# KONSTRUKTION, GÜTE UND KOMPLEXITÄT VON ALGORITHMEN FÜR ABLAUFPLANUNGSPROBLEME

Inauguraldissertation
zur Erlangung des akademischen Grades
eines Doktors der Wirtschaftswissenschaften
(doctor rerum oeconomicarum)
der Rechts- und Wirtschaftswissenschaftlichen Fakultät
der Universität des Saarlandes

vorgelegt von

Klaus Huckert

aus Merzig-Harlingen

Saarbrücken 1979

# KONSTRUKTION, GÜTE UND KOMPLEXITÄT VON ALGORITHMEN FÜR ABLAUFPLANUNGSPROBLEME

Inauguraldissertation
zur Erlangung des akademischen Grades
eines Doktors der Wirtschaftswissenschaften
(doctor rerum oeconomicarum)
der Rechts- und Wirtschaftswissenschaftlichen Fakultät
der Universität des Saarlandes

vorgelegt von

Klaus Huckert

aus Merzig-Harlingen

Saarbrücken 1979

UNIV.
LEIDEN
BIBL.

Tag der Disputation: 21.12.1979
Dekan: Professor Dr. B. Tietz
Erstberichterstatter: Professor Dr. W. Dinkelbach
Zweitberichterstatter: Professor Dr. A.-W. Scheer

Dem Andenken meiner Mutter

Die vorliegende Arbeit wurde im Wintersemester 1979/80 als Dissertation an der Rechts- und Wirtschaftswissenschaftlichen Fakultät der Universität des Saarlandes angenommen.

Herrn Professor Dr. W. Dinkelbach möchte ich für die Unterstützung, die er dieser Arbeit zuteil werden ließ, herzlich danken. Herrn Professor Dr. A.-W. Scheer bin ich für die Übernahme des Korreferates dankbar.

Mit den Herren R. Dupré, J. Jahn, R. Rhode, O. Roglin, O. Rosenberg und R. Weber konnte ich Teile der Arbeit diskutieren, auch Ihnen gilt mein Dank. Gleichermaßen danke ich Frau Barbara Bopp dafür, daß sie die vorliegende Fassung der Dissertation sorgfältig geschrieben hat.

## Inhaltsübersicht


| | | Seite |
|---|---|---|
| 0. | Einleitung | 1 - 3 |
| 1. | Grundlagen und Problemstellung | 4 |
| 1.1. | Definition eines Entscheidungsproblems für Job-Shop-Probleme | 5 - 7 |
| 1.2. | Zielsetzungen in der Ablaufplanung | 8 |
| 1.2.1. | Minimierung der Gesamtdurchlaufzeit | 8 - 9 |
| 1.2.2. | Minimierung der Zykluszeit | 9 |
| 1.2.3. | Minimierung der Gesamtbelegungszeit | 9 - 10 |
| 1.3. | Eine entscheidungstheoretische Interpretation von Zielsetzungen in der Ablaufplanung | 10 - 15 |
| 1.4. | Ablaufplanungsprobleme als disjunktive Graphen | 16 - 18 |
| 2. | Zur Effizienz von Algorithmen | 19 - 20 |
| 2.1. | Grundlegende Begriffe der Komplexitätstheorie | 20 - 30 |
| 3. | Algorithmen zur Erzeugung zulässiger Ablaufpläne | 31 |
| 3.1. | Zulässige Ablaufpläne | 31 - 38 |
| 3.2. | Semiaktive Ablaufpläne | 38 - 43 |
| 3.3. | Aktive Ablaufpläne | 43 - 49 |
| 3.4. | Unverzögerte Ablaufpläne | 49 - 54 |
| 3.5. | Zusammenhänge zwischen semiaktiven, aktiven und unverzögerten Ablaufplänen | 54 - 56 |
| 3.6. | Untere Schranken für optimale Zielfunktionswerte | 57 - 72 |

Seite


| | | |
|---|---|---|
| 4. | Gemischt-ganzzahlige Ablaufplanungsmodelle und ihre Lösungsmöglichkeiten | 73 |
| 4.1. | Das MANNE-Modell | 73 - 74 |
| 4.1.1. | Das MANNE-Modell für die Zielsetzung „Minimierung der Gesamtdurchlaufzeit" | 75 - 77 |
| 4.1.2. | Das MANNE-Modell für die Zielsetzung „Minimierung der Zykluszeit" | 78 - 80 |
| 4.1.3. | Das MANNE-Modell für die Zielsetzung „Minimierung der Belegungszeit" | 80 - 83 |
| 4.2. | Die Benderssche Partitionsmethode zur Lösung gemischt-ganzzahliger Entscheidungsmodelle | 84 - 88 |
| 4.3. | Die Anwendung der Bendersschen Partitionsmethode auf Ablaufplanungsprobleme | 88 - 119 |
| 5. | Branch und Bound-Methoden für Ablaufplanungsprobleme | 120 |
| 5.1. | Das Konzept von Branch und Bound-Algorithmen | 120 - 124 |
| 5.2. | Der Branch und Bound-Algorithmus von GREENBERG | 124 - 141 |
| 6. | Numerische Untersuchungen | 142 |
| 6.1. | Numerische Ergebnisse mit der Bendersschen Partitionsmethode | 142 - 149 |
| 6.2. | Numerische Ergebnisse mit semiaktiven, aktiven und unverzögerten Ablaufplänen | 149 - 159 |
| Anhang 1 | | 160 - 190 |
| Anhang 2 | | 191 - 208 |
| Literaturverzeichnis | | 209 - 211 |

## 0. Einleitung

Ablaufplanungsprobleme spielen in unterschiedlichen Wissenschaftsgebieten eine bedeutende Rolle. In der Betriebswirtschaftslehre sind im Rahmen der Produktionsplanung Ablaufplanungsprobleme zu lösen, die darin bestehen, eine Menge von Aufträgen auf Maschinen so einzuplanen, daß eine bestimmte Zielsetzung optimiert wird. Im Gebiet der Informatik sind im Bereich der Betriebssysteme Ablaufplanungsprobleme zu finden. Stehen eine Reihe von Programmen zur Bearbeitung an, die auf unterschiedlichen Stufen bearbeitet werden müssen, so stellt sich auch hier die Frage nach der Reihenfolge, in der diese Verarbeitungen durchgeführt werden sollen. Interessant ist die Tatsache, daß lange Zeit die Bearbeitung von Reihenfolgeproblemen in der Informatik und Betriebswirtschaftslehre nebeneinanderher verlief, ohne daß zwischen beiden Wissenschaftsgebieten ein Erfahrungsaustausch stattgefunden hätte. Erst seit etwa 1970 ist zu beobachten, wie beide Wissenschaftsgebiete ihre Ergebnisse und Probleme miteinander austauschen. Gemeinsame Grundlage der beiden Gebiete ist die Algorithmentheorie, d.h. die Theorie zur numerischen Lösung von Entscheidungsproblemen. Etwa zwanzig Jahre lang wurde für unterschiedliche Reihenfolgeprobleme (im Bereich der Informatik heißen sie auch Scheduling-Probleme) versucht, numerische Verfahren oder Algorithmen anzugeben, die nach Möglichkeit relativ rasch die gewünschte Lösung liefern sollten.

In den meisten Fällen war das Ergebnis dieser Bemühungen entmutigend. Nur für eine geringe Anzahl von Scheduling-Problemen konnten „gutartige" Algorithmen gefunden werden. Bei den anderen Problemen konnten nur

Algorithmen angegeben werden, deren Rechenumfang exponentiell mit der Größe der Probleme anstieg. Bahnbrechende Arbeiten von COOK und KARP gaben die theoretische Begründung, warum so viele diskrete Entscheidungsprobleme - also auch Ablaufplanungsprobleme - sich vermutlich niemals effizient exakt lösen lassen. Auch ohne die Kenntnis dieser theoretischen Begründungen wurde versucht, mit Hilfe von heuristischen Verfahren, die nicht unbedingt die optimale Lösung liefern, Ablaufplanungsprobleme numerisch zu lösen.

Es ist nun das Anliegen dieser Arbeit, exakte und heuristische Verfahren zur Lösung von Scheduling-Problemen darzustellen und die numerischen Eigenschaften (Laufzeit der Algorithmen, Güte der gewonnenen Ergebnisse) zu vergleichen und zu diskutieren.

In Kapitel 1 der vorliegenden Arbeit wird das betriebswirtschaftliche Ablaufplanungsproblem erläutert und die Prämissen dargestellt, unter denen das Problem erörtert werden soll. Des weiteren werden drei mögliche Zielsetzungen entscheidungstheoretisch hergeleitet, um mögliche Zusammenhänge aufzuzeigen. Da sehr viele Lösungsverfahren graphentheoretische Grundlagen benutzen, wird das Scheduling-Problem auch mit disjunktiven Graphen definiert.

Kapitel 2 erläutert grundlegende Begriffe aus der Komplexitätstheorie für Algorithmen. Diese Begriffe spielen für die folgenden Ausführungen eine wesentliche Rolle. Außerdem werden kurz Ergebnisse über die numerische Lösbarkeit von Ablaufplanungsproblemen angegeben.

Algorithmen zur Erzeugung von Ablaufplänen werden in Kapitel 3 hinsichtlich ihres Rechenaufwands und ihrer Güte untersucht. Dabei wird versucht, untere und obere Schranken für die gesuchten optimalen Zielwerte anzugeben.

Ein gemischt-ganzzahliges Entscheidungsmodell von MANNE wird in Kapitel 4 vorgestellt und graphentheoretisch erläutert. Danach wird der Benderssche Partitionsalgorithmus zur Lösung gemischt-ganzzahliger Entscheidungsmodelle auf Ablaufplanungsprobleme spezialisiert.

In Kapitel 5 werden Grundmodelle für Branch und Bound-Methoden vorgestellt, die zur Lösung von Scheduling-Problemen geeignet sind.

Das letzte Kapitel der vorliegenden Arbeit beschäftigt sich mit numerischen Untersuchungen. Dabei werden die Ergebnisse, die mit den Computerprogrammen aus Anhang 1 und Anhang 2 gefunden wurden, vorgestellt.

## 1. Grundlagen und Problemstellung

Die Produktionsplanung eines Unternehmens umfaßt neben der Produktionsprogramm- und Bereitstellungsplanung die Planung des Produktionsprozesses. Wesentlicher Teil der Prozeßplanung ist die Ablaufplanung. Die im Rahmen der Ablaufplanung zu lösenden Probleme sind weitgehend abhängig von der Art der Fertigung. Ein Problemtyp läßt sich dadurch kennzeichnen, daß zu Beginn der Planungsperiode (dies sei im folgenden immer der Zeitpunkt Null) N verschiedene Aufträge (Jobs) auf M unterschiedlichen Maschinen zu bearbeiten sind. Die Reihenfolge der auf den M unterschiedlichen Maschinen durchzuführenden Bearbeitungen eines Auftrags kann technologisch festgelegt sein. Ist die Reihenfolge der Bearbeitungen nicht für alle Aufträge gleich, so spricht man von Werkstattfertigung (*Job-Shop-Scheduling*), anderenfalls von Reihenfertigung (*Flow-Shop-Scheduling*). Die Aufgabe der Ablaufplanung besteht nun darin, die in bezug auf eine vorgegebene Zielsetzung optimale Reihenfolge der N Aufträge auf den M Maschinen zu bestimmen. Im folgenden wird das Ablaufplanungsproblem als Job-Shop-Problem bezeichnet, da unter diesem Begriff auch das Flow-Shop-Problem subsumiert werden kann. Die klassischen Ansätze zur Lösung dieses Job-Shop-Problems setzen eine Reihe von Prämissen voraus. So müssen etwa Bearbeitungsdauern und technologische Reihenfolgebedingungen für jeden Auftrag bekannt sein. Weiterhin wird angenommen, daß eine Maschine zur gleichen Zeit nur einen Auftrag bearbeiten kann und daß eine angefangene Bearbeitung eines Auftrags ohne Unterbrechung zu Ende geführt wird. Für eine umfangreiche Diskussion der Prämissen bei Ablaufplanungsproblemen sei z.B. auf SEELBACH [54] und SIEGEL [55] verwiesen.

## 1.1. Definition eines Entscheidungsproblems für Job-Shop-Probleme

Beim allgemeinen Job-Shop-Problem wird von N Aufträgen ausgegangen, die auf M unterschiedlichen Maschinen zu bearbeiten sind. Die Bearbeitungszeiten der einzelnen Aufträge auf den M Maschinen sowie die Reihenfolgen, in denen die Aufträge auf den einzelnen Maschinen bearbeitet werden, sind als Daten vorgegeben.

In Zukunft werden die folgenden Bezeichnungen verwendet (vgl. z.B. DINKELBACH [18], RINNOOY KAN [51], SEELBACH [54]):

**Definition 1.1:**

$p_{IJ}$ := gegebene Bearbeitungsdauer des Auftrags I auf Maschine J, wobei $p_{IJ} \geq 0$ (I = 1,...,N; J = 1,...,M); die N*M Zahlen $p_{IJ}$ werden zu der Matrix P der Bearbeitungszeiten

$$P = \begin{pmatrix} p_{11} & \cdots & p_{1M} \\ \vdots & & \vdots \\ p_{N1} & \cdots & p_{NM} \end{pmatrix}$$

zusammengefaßt.

$q_{I(M)}$ := $([1]_I,\ldots,[M]_I)$ sei die gegebene Maschinenfolge für Auftrag I (I = 1,...,N); $q_{I(M)}$ ist eine Permutation der Zahlen 1 bis M; die im allgemeinen unterschiedlichen Maschinenfolgen für alle Aufträge werden durch die Maschinenfolgematrix Q

$$Q = \begin{pmatrix} [1]_1 & \cdots & [M]_1 \\ \vdots & & \vdots \\ [1]_N & \cdots & [M]_N \end{pmatrix}$$

dargestellt.

$h_{IJ}$ := Beginnzeitpunkt der Bearbeitung des Auftrags I auf der Maschine J (I = 1,...,N; J = 1,...,M); die N*M Beginnzeitpunkte $h_{IJ}$ werden zu der Matrix H

$$H = \begin{pmatrix} h_{11} & \cdots & h_{1M} \\ \vdots & & \vdots \\ h_{N1} & \cdots & h_{NM} \end{pmatrix}$$

zusammengefaßt.

$h_{I[J]}$ := Beginnzeitpunkt der Bearbeitung des Auftrags I auf *der* Maschine, die an J-ter Stelle der Maschinenfolge für den Auftrag I tätig wird (I = 1,...,N; J = 1,...,M).

$p_{I[J]}$ := gegebene Bearbeitungsdauer des Auftrags I auf *der* Maschine, die an J-ter Stelle der Maschinenfolge für Auftrag I tätig wird (I = 1,...,N; J = 1,...,M).

Bemerkung 1.1:

Ist die Matrix Q der technologischen Reihenfolgen gegeben, so geht $h_{I[J]}$ in $h_{IK}$ mit K $\in$ {1,...,M} über.

Beispiel 1.1:

$$Q = \begin{pmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \\ 2 & 1 & 3 \end{pmatrix}$$

Betrachtet man z.B. Auftrag 2, so gilt:

$h_{2[1]} := h_{23}$ $\qquad$ $p_{2[1]} := p_{23}$

$h_{2[2]} := h_{22}$ $\qquad$ $p_{2[2]} := p_{22}$

$h_{2[3]} := h_{21}$ $\qquad$ $p_{2[3]} := p_{21}$.

Die Zeitpunkte $h_{IJ}$ (I = 1,...,N; J = 1,...,M) sind Entscheidungsvariablen.

Definition 1.2:

Eine Matrix H heißt *zulässiger* Ablaufplan bezüglich eines Job-Shop-Problems mit N Aufträgen und M Maschinen, wenn

i. alle N*M Beginnzeitpunkte nichtnegativ sind,
ii. die gegebenen Maschinenfolgen für alle Aufträge eingehalten werden,
iii. keine Maschine gleichzeitig mehrere Aufträge zu bearbeiten hat und
iv. eine angefangene Bearbeitung eines Auftrags auf einer Maschine ohne Unterbrechung zu Ende geführt wird.

Die Menge X aller zulässigen Ablaufpläne für dieses Job-Shop-Problem kann also formal dann so dargestellt werden:

$$X := \{H = ((h_{IJ})) \mid h_{IJ} \geq 0 \ (I = 1,\ldots,N;\ J = 1,\ldots,M)$$

$$h_{I[J]} + p_{I[J]} \leq h_{I[J+1]} \quad (I = 1,\ldots,N;\ J = 1,\ldots,M-1)$$

$$h_{LJ} \notin \{y \mid h_{IJ} \leq y \leq h_{IJ} + p_{IJ}\}$$
$$(I = 1,\ldots,N;\ J = 1,\ldots,M;\ L = 1,\ldots,N;\ I \neq L)\}$$

Aus der Alternativenmenge X ist eine Alternative H so zu bestimmen, daß eine vorgegebene Zielsetzung extremiert wird.

Für Ablaufplanungsprobleme lassen sich die meisten Zielfunktionen auf die Minimierung von bestimmten Zielgrößen aus der Matrix H zurückführen.

## 1.2. Zielsetzungen in der Ablaufplanung

In der Literatur wird zur Lösung von Ablaufplanungsmodellen eine Vielfalt von Zielkriterien verwendet. Eine ausführliche Diskussion über ökonomische Prämissen und Implikationen gibt SIEGEL in [55].

**Definition 1.3:**
(Vgl. z.B. BAKER [4], BRUCKER [12], CONWAY/MAXWELL/MILLER [15], SEELBACH [54])
Eine Funktion $f : X \to \mathbb{R}^+$ heißt *reguläre* Zielsetzung für Ablaufplanungsprobleme, wenn

i. f minimiert werden soll,

ii. f eine monoton steigende Abbildung ist, d.h.
falls aus $h^*_{IJ} \geq h_{IJ}$ für alle Indizes IJ
folgt: $f(H^*) \geq f(H)$.

Im folgenden werden drei reguläre Zielsetzungen untersucht, und die Zusammenhänge zwischen ihnen diskutiert.

### 1.2.1. Minimierung der Gesamtdurchlaufzeit

Das am häufigsten benutzte Zielkriterium in der Ablaufplanung ist die Minimierung von Fertigstellungszeitpunkten oder gleichbedeutend damit, die Minimierung von Durchlaufzeiten. Allgemein läßt sich die Durchlaufzeit als diejenige Zeitspanne definieren, in der ein Auftrag im Betrieb (Werkstatt) ist. Die Durchlaufzeit eines Auftrags kann als die Summe aus seinen Bearbeitungszeiten und Wartezeiten betrachtet werden. Die Wartezeit eines Auftrags I vor Maschine J ist die Differenz aus tatsächlichem Starttermin und dem Verfügbarkeitstermin. Dies ist derjenige Zeitpunkt, an dem die technologischen Vorgängeroperationen abgeschlossen sind. Wird mit $w_{IJ}$ die Wartezeit des Auftrags I auf Maschine J bezeichnet, so läßt sich die Durchlaufzeit schreiben:

$$(1\text{-}1) \qquad GD = \sum_{I=1}^{N} \sum_{J=1}^{M} (w_{IJ} + p_{IJ})$$

Da die Summe der Gesamtbearbeitungszeiten eine Konstante ist, entspricht also die Minimierung der Gesamtdurchlaufzeit GD der Minimierung der Gesamtwartezeit.

Da die Durchlaufzeitminimierung aber gleichbedeutend mit der Minimierung der Fertigstellungszeitpunkte ist, läßt sich GD schreiben als

$$(1\text{-}2) \qquad GD = \sum_{I=1}^{N} (h_{I[M]} + p_{I[M]}).$$

### 1.2.2. Minimierung der Zykluszeit

Als Zykluszeit wird diejenige Zeitspanne verstanden, die zwischen dem Eintreffen aller Aufträge und der Beendigung der letzten Bearbeitung am gesamten Auftragsbestand vergeht. Hier wird vorausgesetzt, daß alle Aufträge zu Beginn der Planungsperiode (zum Planungszeitpunkt Null) vorliegen. Dann läßt sich die Zykluszeit Z so berechnen:

$$(1\text{-}3) \qquad Z := \max \{h_{1[M]} + p_{1[M]}, \ldots, h_{N[M]} + p_{N[M]}\}.$$

### 1.2.3. Minimierung der Gesamtbelegungszeit

Als Gesamtbelegungszeit GB aller M Maschinen wird die Summe aller Leerzeiten $l_{IJ}$ und aller Bearbeitungszeiten $p_{IJ}$ bezeichnet. Die Leerzeit $l_{IJ}$ ist dabei die Zeitspanne, in der Maschine J vor Beginn der Bearbeitung des Auftrags I untätig ist.

$$(1\text{-}4) \qquad GB := \sum_{I=1}^{N} \sum_{J=1}^{M} (l_{IJ} + p_{IJ})$$

Da die Summe der Bearbeitungszeiten konstant ist, kann die Minimierung der Gesamtbelegungszeit auf die Mini-

mierung der Gesamtleerzeit zurückgeführt werden. (1-4) kann auch mit Hilfe der Endtermine $h_{IJ} + p_{IJ}$ ($I = 1,\ldots,N$; $J = 1,\ldots,M$) angegeben werden. Die Belegungszeit einer Maschine J ist definiert als das Maximum der Endtermine $h_{IJ} + p_{IJ}$ ($I = 1,\ldots,N$; $J \in \{1,\ldots,M\}$). Mit dieser Überlegung kann (1-4) so geschrieben werden:

$$(1\text{-}5) \qquad GB := \sum_{J=1}^{M} \max \{h_{1J} + p_{1J},\ldots,h_{NJ} + p_{NJ}\}.$$

## 1.3. Eine entscheidungstheoretische Interpretation von Zielsetzungen in der Ablaufplanung

In der Betriebswirtschaftslehre hat seit der Diskussion über Ablaufplanungsprobleme die Frage nach möglichen Zielsetzungen und Zusammenhängen zwischen einzelnen Zielfunktionen ein starkes Interesse gefunden. GUTENBERG diskutiert beispielsweise in [27, S. 151] das Dilemma der Ablaufplanung, das darin besteht, zwei entgegengesetzte Zielsetzungen miteinander in Einklang zu bringen, nämlich auf der einen Seite Wartezeiten für die einzelnen Aufträge und auf der anderen Seite Leerzeiten von Maschinen zu minimieren.

Zusammenfassende Darstellungen über mögliche Zielsetzungen werden in den Arbeiten von ASHOUR [3, S. 34-64], HOSS [29, S. 36-45], RINNOOY KAN [51, S. 16-28] und SEELBACH [54, S. 32-39] diskutiert.

DINKELBACH leitet in [18] mit Hilfe der Theorie der mehrfachen Zielsetzung Aussagen über die Ziele der Minimierung der Durchlaufzeit einerseits und der Minimierung der Zykluszeit andererseits her. Dieser entscheidungstheoretische Ansatz von DINKELBACH wird hier benutzt, um Zusammenhänge und Gegenläufigkeiten der in Abschnitt 1.2 vorgestellten möglichen Zielfunktionen herzuleiten.

In Abschnitt 1.2 wurde erwähnt, daß für Ablaufplanungsprobleme die meisten Zielkriterien sich auf die Minimierung von bestimmten Zeitgrößen aus der Matrix H zurückführen lassen. Formal soll dieses Entscheidungsproblem auf die gleichzeitige Minimierung eines Entscheidungsmodells mit N*M Zielfunktionen zurückgeführt werden.

Definition 1.4:
Das Entscheidungsproblem

(1-6) „min" {H + P | H ∈ X} heißt Matrixminimumproblem.

Bemerkung: (1-6) besitzt N*M Zielfunktionen. Setzt man entweder N oder M gleich Eins, so geht (1-6) in ein Vektorminimumproblem über; wird N und M gleich Eins gesetzt, so stellt (1-6) ein Minimierungsproblem mit einer Zielfunktion dar.
Die Anführungsstriche bei (1-6) sollen daraufhin hindeuten, daß es sich hier nicht um ein gewöhnliches Minimierungsproblem mit einer skalarwertigen Zielfunktion handelt.

Definition 1.5:
(GALE/KUHN/TUCKER [23])
H sei zulässige Lösung von (1-6) im Sinne von Definition 1.2. H heißt dann effiziente oder nicht dominierte Lösung für (1-6), falls keine Matrix H' existiert, so daß

$$h'_{IJ} \leqq h_{IJ} \quad (I = 1,\ldots,N;\ J = 1,\ldots,M)$$

und

$$h'_{IJ} < h_{IJ} \quad \text{(für mindestens einen Index IJ).}$$

Entscheidungsproblem (1-6) kann dann so interpretiert werden, daß alle effizienten Matrizen H ∈ X zu berechnen sind.

Die Menge X der zulässigen Ablaufpläne H besitzt unendlich viele Elemente. Die Mächtigkeit der Menge der effizienten Ablaufpläne H ist im allgemeinen auch unendlich.

Da man bei der Lösung von Ablaufplanungsproblemen nach *einer* optimalen Alternative sucht, muß das Matrixminimumproblem (1-6) mit Hilfe von Ersatz- und Kompromißprogrammen transformiert werden, um eine Lösung zu bestimmen. Kompromißprogramme für Matrixminimumprobleme können aber nur dann sinnvoll sein, wenn ihre Lösung eine effiziente Lösung liefert.

DINKELBACH gibt in [18] Beispiele von Kompromißprogrammen an, die auf der Minimierung des Abstandes zu einem idealen (optimalen), aber nicht erreichbaren Punkt beruhen.

Für das Entscheidungsproblem (1-6) ist die folgende Abstandsfunktion (1-7) ein Kompromißprogramm.

$$\min \{ || H + P - H^* ||_p \mid H \in X\}$$

$$(1\text{-}7) \quad = \min \{ ( \sum_{I=1}^{N} \sum_{J=1}^{M} | h_{IJ} + p_{IJ} - h^*_{IJ} |^p )^{1/p} \mid H \in X\}$$

$H^*$ stellt dabei einen Ablaufplan dar, dessen numerische Werte i.a. nicht erreichbar sind. Beispielsweise kann $H^*$ so aussehen, daß die Werte $h^*_{IJ}$ den frühesten Startterminen zuzüglich der Bearbeitungszeiten entsprechen, d.h., es werden nur die technologischen Reihenfolgebedingungen beachtet, aber nicht die Bedingung, daß eine Maschine nur jeweils einen Auftrag bearbeiten kann. Es gilt also allgemein:

$$(1\text{-}8) \quad h^*_{IJ} \leq h_{IJ} + p_{IJ} \quad \forall \; IJ \quad \text{und} \quad \forall \; H \in X.$$

Angenommen ein Ablaufplaner möchte die Fertigstellungstermine aller N Aufträge gleichzeitig minimieren. Dann erhält das Matrixminimumproblem (1-6) die Form:

$$(1\text{-}9) \quad \text{„min"} \left\{ \begin{pmatrix} h_{1[M]} + p_{1[M]} \\ \vdots \\ h_{N[M]} + p_{N[M]} \end{pmatrix} \mid H \in X \right\}.$$

Kompromißprogramm (1-7) erhält bezüglich des Vektorminimumproblems (1-9) die Form

$$(1\text{-}10) \quad \min \{ ( \sum_{I=1}^{N} \mid h_{I[M]} + p_{I[M]} - h^*_{I[M]} \mid^p )^{1/p} \mid H \in X \}.$$

Wird p = 1 gesetzt (vgl. DINKELBACH [18]), so kann (1-10) geschrieben werden als

$$(1\text{-}11) \quad \min \{ ( \sum_{I=1}^{N} \mid h_{I[M]} + p_{I[M]} - h^*_{I[M]} \mid ) \mid H \in X \}.$$

Wegen (1-8) ist (1-11) gleichbedeutend mit

$$\begin{aligned} & \min \{ \sum_{I=1}^{N} (h_{I[M]} + p_{I[M]} - h^*_{I[M]}) \mid H \in X \} \\ = & \min \{ \sum_{I=1}^{N} h_{I[M]} + \sum_{I=1}^{N} (p_{I[M]} - h^*_{I[M]}) \mid H \in X \} \\ (1\text{-}12) = & \min \{ \sum_{I=1}^{N} h_{I[M]} \mid H \in X \} - \text{const.} \end{aligned}$$

Die Zielfunktion des Kompromißprogramms (1-12) unterscheidet sich in der Form von (1-2) nur durch eine additive Konstante, d.h., die Entscheidungsmodelle (1-12) und (1-2) liefern den gleichen optimalen Ablaufplan H.

Wird bei (1-10) $p = \infty$ (vgl. DINKELBACH [18]) und $H^*$ gleich der Nullmatrix gesetzt, so gilt

$$(1\text{-}13)\quad \lim_{p\to\infty} \min \{ \Big( \sum_{I=1}^{N} \mid h_{I[M]} + p_{I[M]} \mid^{p} \Big)^{1/p} \mid H \in X \}$$

$$(1\text{-}14) = \min \{ \max \{ h_{1[M]}+p_{1[M]}, h_{2[M]}+p_{2[M]}, \ldots, h_{N[M]} + p_{N[M]} \} \mid H \in X \}.$$

Die Zielfunktion des Kompromißprogramms (1-14) unterscheidet sich von der Form der Zielfunktion (1-3) nicht; die Entscheidungsmodelle (1-14) und (1-3) liefern den gleichen optimalen Ablaufplan.
Es sei nun angenommen, ein Ablaufplaner möchte gleichzeitig die Fertigstellungszeitpunkte aller N Aufträge auf den einzelnen Maschinen und die Summe aller Endtermine minimieren, in der die Maschinen belegt sind.
Das Matrixminimumproblem (1-6) nimmt dann die Form an:

$$(1\text{-}15)\quad \text{"min"} \begin{pmatrix} h_{11} + p_{11} \\ \vdots \\ h_{N1} + p_{N1} \end{pmatrix} + \ldots + \text{"min"} \begin{pmatrix} h_{1M} + p_{1M} \\ \vdots \\ h_{NM} + p_{NM} \end{pmatrix}$$

$$(1\text{-}16)\quad = \sum_{J=1}^{M} \text{"min"} \begin{pmatrix} h_{1J} + p_{1J} \\ \vdots \\ h_{NJ} + p_{NJ} \end{pmatrix}.$$

Setzt man im Kompromißprogramm (1-7) $H^*$ gleich der Nullmatrix, so nimmt Kompromißprogramm (1-7) bezüglich des Vektorproblems (1-16) die Form

$$(1\text{-}17)\quad \min \{ \Big( \sum_{J=1}^{M} \sum_{I=1}^{N} \mid h_{IJ} + p_{IJ} \mid^{p} \Big)^{1/p} \mid H \in X \}$$

an.

Wird $p = \infty$ gesetzt, so ergibt sich für (1-17)

$$(1\text{-}18) \quad \lim_{p\to\infty} \min \{ \Big( \sum_{J=1}^{M} \sum_{I=1}^{N} | h_{IJ} + p_{IJ} |^{p} \Big)^{1/p} \mid H \in X \}$$

$$(1\text{-}19) = \min \{ \sum_{J=1}^{M} \max \{ h_{1J} + p_{1J}, \ldots, h_{NJ} + p_{NJ} \} \mid H \in X \}$$

Entscheidungsmodell (1-19) ist äquivalent mit (1-5).

In diesem Abschnitt wurden die Zielfunktionen „Minimierung der Durchlaufzeit", „Minimierung der Zykluszeit" und „Minimierung der Belegungszeit" als verschiedene Arten von Kompromißfunktionen von Matrixminimierungsproblemen hergeleitet. Daraus kann aber geschlossen werden, daß die drei erwähnten Zielsetzungen im allgemeinen zu unterschiedlichen optimalen Ablaufplänen H ∈ X führen. DINKELBACH schreibt in [18, S. 564]: „Jede optimale Alternative eines Kompromißprogramms eines Vektoroptimierungsproblems hat die Eigenschaft, nur im Hinblick auf die vom Entscheidungsträger gewählte Kompromißzielfunktion optimal (kompromißoptimal) zu sein, wobei verschiedene Kompromißzielfunktionen verschiedene Präferenzvorstellungen des Entscheidungsträgers widerspiegeln".

Die hier gegebene entscheidungstheoretische Interpretation von verschiedenen Zielsetzungen als spezielle Kompromißprogramme von Matrixminimumproblemen bestätigen also die Unterschiedlichkeit der diskutierten Zielsetzungen.

## 1.4. Ablaufplanungsprobleme als disjunktive Graphen

An dieser Stelle soll das Ablaufplanungsproblem mit *disjunktiven* Graphen definiert werden. Diese Definitionen gehen auf ROY und SUSSMANN [52] zurück. Sehr viele Lösungsverfahren für Scheduling Probleme beruhen auf dieser Darstellung.

**Definition 1.4:**

Ein *disjunktiver Graph* G ist ein Quadrupel der Mengen V, C, D und P, wobei

- V die Knotenmenge der durchzuführenden N*M Bearbeitungen ist. Jede Bearbeitung läßt sich eindeutig durch einen Index IJ (I = 1,...,N; J = 1,...,M) kennzeichnen,
- $C \subset V \times V$ die Menge der *konjunktiven* Pfeile ist, die die technologischen Reihenfolgebedingungen für die einzelnen Aufträge wiedergeben,
- $D \subset V \times V$ die Menge der *disjunktiven* Pfeile ist, die die möglichen Auftragsfolgen für die einzelnen Maschinen darstellen,
- P die Matrix der Bearbeitungszeit ist. Aus P wird die Kantenbewertung für die Pfeile aus C und D konstruiert.

**Beispiel 1.2:**

Gegeben seien drei Aufträge, die auf drei unterschiedlichen Maschinen zu bearbeiten sind:

$$P = \begin{pmatrix} 4 & 2 & 1 \\ 3 & 7 & 5 \\ 2 & 4 & 6 \end{pmatrix} \qquad Q = \begin{pmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \\ 1 & 2 & 3 \end{pmatrix}.$$

Der disjunktive Graph G besteht aus den vier Mengen {V,C,D,P} mit

V := {11,12,13,21,22,23,31,32,33}
C := {(11,12),(12,13),(23,22),(22,21),(31,32),(32,33)}
D := {(11,21),(21,11),(11,31),(31,11),(21,31),(31,21),
(12,22),(22,12),(12,32),(32,12),(22,32),(32,22),
(13,23),(23,13),(13,33),(33,13),(23,33),(33,23)}
P := wie angegeben.

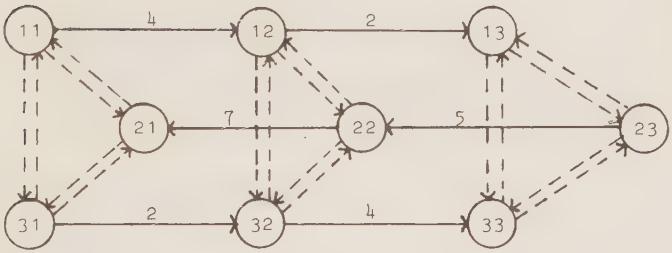



Abbildung 1

Bemerkung 1.2:
Die disjunktiven Pfeile wurden der Übersichtlichkeit halber nicht bewertet, die konjunktiven Pfeile wurden mit den Bearbeitungszeiten bewertet. Desweiteren wird an dieser Stelle nicht darauf eingegangen, wie die verschiedenen Zielsetzungen in diesem Modell dargestellt werden können. Dies wird in Kapitel 4 geschehen.

Aus dem disjunktiven Graphen G läßt sich ein Ablaufgraph F konstruieren.

Definition 1.5:
Ein Ablaufgraph F ist ein Tripel der Mengen V, C ∪ $D^*$ und P, wobei

- V,P dieselben Bedeutungen wie in Definition 1.4 haben,

- C ∪ $D^*$ die Vereinigungsmenge der konjunktiven Pfeile aus C mit der Menge $D^*$ ist, die aus D entsteht, indem jeweils genau ein Paar (IJ,KJ) oder ein Paar (KJ,IJ) aus der möglichen Menge {(IJ,KJ), (KJ,IJ)} ausgewählt wird.

Die Alternativenmenge X aller zulässigen Ablaufpläne läßt sich mit Hilfe der Graphentheorie charakterisieren.

Satz 1.1:
H ist genau dann aus X, wenn der Ablaufgraph F ohne Zyklen ist.
Beweis: ROY und SUSSMANN [52].

Beispiel 1.3:
Für den in Beispiel 1.2 beschriebenen disjunktiven Graphen G kann folgender Ablaufgraph F = {V,C ∪ $D^*$,P} konstruiert werden, wobei
V,P := wie in Beispiel 1.2 angegeben sind,
C ∪ $D^*$ := {(11,12),(12,13),(23,22),(22,21),(31,32),(32,33), (11,21),(11,31),(31,21),(22,12),(32,12),(22,32), (23,13),(33,13),(23,33)} ist.

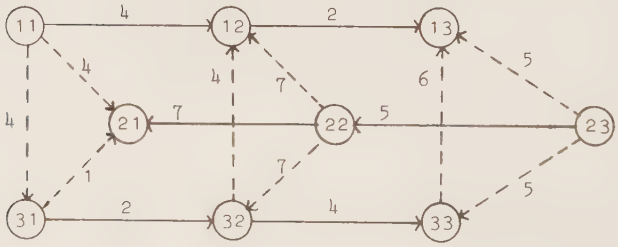



Abbildung 2

Zur Erkennung von Zyklen in Graphen ist beispielsweise das Verfahren von FORD (vgl. z.B. für eine Darstellung [11]) geeignet. Das Verfahren von MARIMONT [45] eignet sich für Scheduling-Probleme aus rechentechnischen Gründen besser. Dieser Algorithmus und sein Rechenaufwand werden im dritten Kapitel erläutert.

## 2. Zur Effizienz von Algorithmen

Um die optimale Alternative eines Entscheidungsmodells zu bestimmen, muß das Entscheidungsmodell numerisch gelöst werden. Zur numerischen Lösung solcher Probleme wurden und werden Algorithmen entwickelt. Algorithmen werden meistens auf Rechenanlagen programmiert, so daß sofort die Frage entsteht, wie „schnell" oder wie „gut" ein Algorithmus arbeitet. Güte und Schnelligkeit hängen mit folgenden Fragestellungen zusammen:

i. Wieviel Rechenzeit benötigt ein Verfahren zur numerischen Lösung eines Entscheidungsmodells?
ii. Wieviel Speicherplatz benötigt der Algorithmus?

Die übliche Vorgehensweise zur Bestimmung des Rechenaufwandes war früher - d.h., bevor die Informatik in Verbindung mit der numerischen Mathematik eine wissenschaftliche Disziplin wurde - folgende: Um ein Entscheidungsproblem numerisch zu lösen, wurde ein Algorithmus entwickelt oder auf bestehende Verfahren zurückgegriffen, dann wurde ein Computerprogramm geschrieben, mit dem einige Beispiele mit verschiedenen Datenkonstellationen für das Entscheidungsmodell durchgerechnet wurden. Als Effizienz dieses Algorithmus wurden dann Rechenzeit und Speicherplatzbedarf für die Testbeispiele genannt. Die Nachteile solchen Vorgehens liegen auf der Hand. Es ist nicht ersichtlich, wie sich ein Algorithmus bei Variation der Daten des Entscheidungsmodells verhält. Außerdem werden bei unterschiedlichen Computern und Betriebssystemen verschiedene Rechenzeiten erzielt. Es existieren in der Literatur zahlreiche Beispiele dafür, wie Rechenzeiten von unterschiedlichen Computern für gleiche oder ähnliche Algorithmen nebeneinandergestellt werden.

Die theoretisch und praktisch interessierende Fragestellung zur Effizienz von Algorithmen lautet heute: Wie verhält sich ein Verfahren bei Eingabe von gewissen Inputgrößen vom Umfang n unabhängig von der benutzten Rechenanlage? Zwei Maßgrößen bieten sich dafür an: Das Verhalten des Algorithmus im schlechtesten Fall (worst-case-Komplexität) und das Verhalten im Mittel (average-case-Komplexität).

In Abhängigkeit der Eingabegrößen vom Umfang n soll die Laufzeit eines Algorithmus im schlechtesten Fall oder im Mittel angegeben werden. Neben der Laufzeit (Zeitkomplexität) kann auch die Speicherplatzkomplexität allgemein untersucht werden. Letzteres wird im Rahmen dieser Arbeit nicht untersucht.

Bei den erwähnten Effizienzmaßen tauchen sofort weitere Fragestellungen auf, z.B. wie die Kodierung der Eingabegrößen erfolgen soll. Auch lassen sich Einwände gegen die beiden Effizienzmaße konstruieren; beispielsweise weiß man, daß das Verhalten des Simplexalgorithmus im schlechtesten Fall ein exponentielles, d.h. über alle Grenzen wachsendes Zeitverhalten ist, wenn die Anzahl der Restriktionen oder der Variablen vergrößert wird. Trotzdem arbeitet der Simplexalgorithmus auch noch mit mehreren zehntausend Variablen und Restriktionen außerordentlich gut.

## 2.1. Grundlegende Begriffe der Komplexitätstheorie

Da Effizienzmaße unabhängig von einer realen Rechenanlage definiert und errechnet werden sollen, muß eine Abstraktion von Rechenanlagen erfolgen. Auf eine exakte und ausführliche Darstellung, die tief in die

Automatentheorie führen würde, soll hier verzichtet werden. Für genauere Ausführungen sei z.B. auf AHO/HOPCROFT/ULLMANN [2] und MEHLHORN [46] verwiesen.

Definition 2.1:

Eine Turingmaschine TM ist ein 7-Tupel

$$(Q, T, I, \delta, b, q_0, q_f),$$

wobei

$Q$ eine endliche Menge von sogenannten Zuständen,
$T$ ein gegebenes Bandalphabet,
$I \subseteq T$ die Menge des Inputalphabetes,
$b \in T \setminus I$ ein Sonderzeichen (beispielsweise das Symbol blank),
$q_0$ der Anfangszustand der Turingmaschine,
$q_f$ der Endzustand der Turingmaschine und
$\delta$ die Übergangsfunktion ist.

Dabei ist $\delta$ eine Abbildung von

$$Q \times T \to Q \times \{T \times \{L, R, S\}\},$$

wobei die Menge $\{L, R, S\}$ anzeigt, ob die Turingmaschine nach links ($L$), nach rechts ($R$) oder überhaupt nicht bewegt wird ($S$).

In Abhängigkeit des Zustandes und des gelesenen Symbols gelangt die TM mit Hilfe von $\delta$ in einen neuen Zustand, druckt ein neues Zeichen und bewegt sich unter Umständen.

Bemerkung 2.1:

Anschaulich kann die TM als Modell eines Rechners mit folgenden Eigenschaften definiert werden:

i. Eine TM besitzt eine Steuereinheit,
ii. einen Lese-Schreibkopf,
iii. ein nach rechts unbeschränktes Rechenband, das in Quadrate eingeteilt ist,

iv. die TM kann Zeichen eines gegebenen Alphabets auf das Rechenband drucken,
v. die TM kann jeweils um ein Quadrat nach rechts, nach links oder aber gar nicht bewegt werden,
vi. die TM kann nur eine endliche Menge von Zuständen annehmen,
vii. die TM kann eine Turingtafel (Programm) beachten.

Rechenanweisungen können also mit Hilfe einer Turingtafel angegeben werden.

Es wird zwischen *deterministischen* und *nichtdeterministischen* TM unterschieden.

Deterministische TM sind durch folgende Besonderheit gekennzeichnet: In Abhängigkeit des Zustandes, in der sich die Turingmaschine befindet, gelangt die Turingmaschine in höchstens *einen* anderen Zustand. Nichtdeterministische Turingmaschinen können in eine Menge von Folgezuständen gelangen. Deterministische TM sind vereinfachte Modelle von real existierenden Computern, während nichtdeterministische Turingmaschinen in der Realität nicht existierender Computer darstellen. Augenscheinlich läßt sich jede deterministische TM als nichtdeterministische darstellen.

Definition 2.2:
Unter der Zeitkomplexität eines Algorithmus soll die Anzahl der Operationen verstanden werden, die eine Turingtafel benötigt, um zur Lösung einer Eingabe vom Umfang n zu gelangen. Die Turingmaschine gelangt in einen sogenannten akzeptierenden Endzustand.

Sehr oft werden Turingmaschinen eingesetzt, um Sprachen zu erkennen. Eine Sprache ist durch eine Menge von Worten über einem endlichen Bandalphabet gekennzeichnet. Jedes Entscheidungsproblem korrespondiert zu einem Sprachenerkennungsproblem, wie Definition 2.3 zeigt.

Definition 2.3:
Ein diskretes Entscheidungsproblem besteht aus folgenden Angaben:

S1 := Menge von Alternativen,
P1 := {s $\in$ S1 | s besitzt eine bestimmte Eigenschaft}
und weiterhin aus der Frage:

Ist P1 = $\emptyset$ oder P1 $\neq \emptyset$?

Aus Definition 2.3 ergibt sich, daß jedes Entscheidungsproblem als Sprache über der Menge {0,1} interpretiert werden kann. Für genauere Angaben über Zusammenhänge zwischen Sprachen, Entscheidungsproblemen und Optimierungsmodellen sei beispielsweise auf AHO/HOPCROFT/ULLMANN [2] und MEHLHORN [46] verwiesen. Insbesondere läßt sich jedes Scheduling-Problem, das bekanntermaßen ein diskretes Entscheidungsproblem ist, als Sprachenerkennungsproblem formulieren.

Beispiel 2.1:
Jedes Scheduling-Problem läßt sich als Sprachenerkennungsproblem für eine Turingmaschine über dem Alphabet {0,1} interpretieren: Wenn eine natürliche Zahl k gegeben ist, so lautet das Scheduling-Problem: Existiert ein zulässiger Ablaufplan H $\in$ X, für den gilt, daß der

Zielfunktionswert des Plans H kleiner oder gleich k ist?

Entscheidungsprobleme können allgemein als Sprachenerkennungsprobleme definiert werden. Die Zeitkomplexität zur Erkennung einer Sprache mit Hilfe einer Turingmaschine kann also über die Anzahl der durchzuführenden Operationen bestimmt werden.

Es sei an dieser Stelle bemerkt, daß es meistens nicht möglich oder zu aufwendig ist, die exakte Anzahl von durchzuführenden Operationen zu bestimmen. Man ist gezwungen, asymptotische Aussagen zu finden und obere Schranken für die Zeitkomplexität von Algorithmen anzugeben.

Definition 2.4:
$f : \mathbb{N}_0 \to \mathbb{N}_0$ sei eine Funktion der Eingabegröße n. Dann bedeutet die Aussage: Die Zeitkomplexität T(n) eines Algorithmus beträgt O(f(n)): Es existiert eine Konstante $c \in \mathbb{R}_+$ und ein gewisses $n_1 \in N$, so daß $T(n) \leq c * f(n)$ für $n \geq n_1$.

Neben der Zeitkomplexität von Algorithmen kann auch die Komplexität von Entscheidungsmodellen interessant sein.

Definition 2.5:
Ein Entscheidungsmodell besitzt höchstens die Komplexität O(f(n)), wenn ein Algorithmus zur Lösung des Entscheidungsmodells existiert, dessen Zeitkomplexität O(f(n)) beträgt.

Zwei Klassen von Entscheidungsproblemen spielen in der Informatik eine besondere Rolle.

Definition 2.6:
$\mathcal{P}$ := {$\mathcal{L}$ | $\mathcal{L}$ = Sprache, für die eine deterministische TM und ein Polynom p mit T(n) $\leq$ O(p(n)) existiert, und die Sprache $\mathcal{L}$ wird von TM akzeptiert}.

Definition 2.7:
$\mathcal{NP}$ := {$\mathcal{L}$ | $\mathcal{L}$ = Sprache, für die eine nichtdeterministische TM und ein Polynom p mit T(n) $\leq$ O(p(n)) existiert, so daß $\mathcal{L}$ von TM akzeptiert wird}

Es ist auf den ersten Blick nicht einsichtig, was diese beiden Klassen $\mathcal{P}$ und $\mathcal{NP}$ mit Entscheidungsproblemen zu tun haben. Überlegt man sich aber, daß jedes Entscheidungsproblem als Sprachenerkennungsproblem darstellbar ist, so wird die Bedeutung von $\mathcal{P}$ und $\mathcal{NP}$ klarer.

Die Klasse $\mathcal{P}$ stellt diejenigen Entscheidungsmodelle dar, für die Algorithmen existieren, die in ihrer Zeitkomplexität polynomiell beschränkt sind und auf einem real existierenden Computer programmiert werden können. Die Klasse $\mathcal{NP}$ enthält neben der Klasse $\mathcal{P}$ diejenigen Entscheidungsmodelle, für die nur auf einem fiktiven Computer Algorithmen existieren, die in ihrer Zeitkomplexität polynomiell beschränkt sind.

Vereinfacht läßt sich die Klasse $\mathcal{P}$ mit der Klasse der „gutartigen" Entscheidungsmodelle und $\mathcal{NP} \setminus \mathcal{P}$ mit der Klasse der „bösartigen" Entscheidungsmodelle identifizieren.

COOK [16] zeigte, daß die Klasse der $\mathcal{NP}$-Probleme noch weiter zerlegt werden kann.

Definition 2.6:
Ein Entscheidungsproblem E1 soll $\mathcal{NP}$-vollständig heißen, wenn gilt:

i. E1 $\in \mathcal{NP}$,

ii. für alle Entscheidungsprobleme E2 aus $\mathcal{NP}$ gilt: Jedes Entscheidungsmodell E2 läßt sich mit einem polynomiell beschränkten Algorithmus auf E1 zurückführen.

Mit dieser Definition allein kann im allgemeinen nicht die $\mathcal{NP}$-Vollständigkeit eines Entscheidungsmodells gezeigt werden. Der nächste Satz von COOK [16] gibt ein konstruktives Mittel zur Hand, wie die $\mathcal{NP}$-Vollständigkeit gezeigt werden kann.

Satz 2.1:
Sei $E_o$ $\mathcal{NP}$-vollständig. Läßt sich $E_o$ in polynomiell beschränkter Zeit auf ein Entscheidungsmodell $E_1$ transformieren, so ist $E_1$ $\mathcal{NP}$-vollständig.

KARP gab in [34] eine Vielzahl von Entscheidungsproblemen an, die zur Klasse der $\mathcal{NP}$-vollständigen Probleme gehören. Darunter sind:

i. Das KNAPSACK-Problem
Gegeben seien positive ganze Zahlen $a_1,\ldots,a_t,b$.
Existiert eine Teilmenge $S1 \subset T = \{1,\ldots,t\}$, so daß
$\sum_{i \in S1} a_i = b$?

Normalerweise wird in der Literatur des Operations Research das binäre Knapsack-Problem so formuliert:

$$(2\text{-}1) \qquad \max \sum_{i=1}^{t} c_i * x_i$$

$$\text{u.d.N.} \sum_{i=1}^{t} a_i * x_i = b$$

$$x_i \in \{0,1\} \quad i = 1,\ldots,t.$$

Es stellt sich die Frage, ob die zwei dargestellten Formulierungen äquivalent sind. Dies kann mit Hilfe der Definition 2.3 gezeigt werden.

Bezeichnet man mit S1 die Menge der 0-1 Tupel $x_1,\ldots,x_t$, die die Nebenbedingung von (2-1) erfüllen, so geht (2-1) in

$$\max_{S1} \{ \sum_{i=1}^{t} c_i * x_i \}. \qquad (2\text{-}2)$$

über. Definiert man mit $P_K$ alle diejenigen 0-1 Tupel $x_1,\ldots,x_t$, für die $\sum_{i=1}^{t} c_i * x_i > K$ (K fix) gilt, so lautet nach Definition 2.3 das Entscheidungsproblem: Entscheide, ob $P_K = \emptyset$ oder $P_K \neq \emptyset$ ist, d.h., existiert in der Menge S1 eine Folge von 0-1 Tupeln, die mit den entsprechenden $a_i$ multipliziert gerade b ergeben. Diese Fragestellung ist aber die zuerst dargestellte Formulierung des Knapsack-Problems. Das Optimierungsproblem (2-1) kann also in eine Folge von Entscheidungsproblemen transformiert werden, indem man K beispielsweise immer um den Wert 1 erhöht.

ii. Binäre Programmierung

Gegeben sei eine ganzzahlige Matrix A und ein ganzzahliger Vektor d. Existiert ein (0,1)-Vektor a mit $A * a \geq d$.

Interessant sind auch Probleme, die in der Klasse $\mathcal{P}$ liegen. Zu diesen Problem gehören (vgl. z.B. AHO/HOPCROFT/ULLMANN [2], KNUTH [35], MEHLHORN [46]):

i. Die Matrizenmultiplikation: die Standardmethode, zwei n×n Matrizen miteinander zu multiplizieren,

besitzt die Zeitkomplexität $O(n^3)$.

ii. Das Lösen von linearen Gleichungssystemen der Form A * x = b, wobei A(n×n),x,b (n×1)-Vektor ist. Dies kann mit Hilfe einer sogenannten LU-Technik in einer Zeitkomplexität von $O(n^3)$ geschehen.

iii. Die Bestimmung kürzester (längster) Wege in Graphen. Der Algorithmus von DIJKSTRA [17] leistet dies in $O(n^2)$[1], wobei n die Anzahl der Ecken des Graphen ist (vgl. NOLTEMEIER [50, S. 105].

iv. Die Bestimmung des maximalen Elementes einer n-elementigen Menge. Durch paarweisen Vergleich kann es in O(n) bestimmt werden.

In der Ablaufplanung wird seit etwa 30 Jahren versucht, für Scheduling-Probleme effiziente, d.h. polynomiell beschränkte Algorithmen anzugeben. Nur für einige spezielle Probleme konnten effiziente Algorithmen angegeben werden, so wurde z.B. von JOHNSON in [33] ein Algorithmus entwickelt, der das folgende Scheduling-Problem in O(N*log N) löst, wobei N die Anzahl der Aufträge und log N der Logarithmus von N zur Basis 2 ist.

Gegeben seien N Aufträge, die auf zwei Maschinen zu bearbeiten sind. Alle Aufträge sollen zuerst auf Maschine 1 und dann auf Maschine 2 bearbeitet werden. Minimiert werden soll die Gesamtdurchlaufzeit dieser N Aufträge.

Viele Autoren beschäftigten sich mit der numerischen Lösung von Job-Shop-Problemen. CONWAY/MAXWELL/MILLER beschreiben in [15, S. 103] die Problematik der exakten Lösung von Ablaufplanungsproblemen in folgender Weise:

---

1) Auf die Problematik, wie Graphen und ihre Nachfolgebeziehungen kodiert werden, wird hier nicht eingegangen. Vgl. zu dieser Frage MEHLHORN [46, S. 142 f.].

"MANY PROFICIENT PEOPLE HAVE CONSIDERED THE PROBLEM AND ALL HAVE COME AWAY ESSENTIALLY EMPTY-HANDED. SINCE THIS FRUSTRATION IS NOT REPORTED IN THE LITERATURE, THE PROBLEM CONTINUES TO ATTRACT INVESTIGATORS, WHO JUST CANNOT BELIEVE THAT A PROBLEM SO SIMPLY STRUCTURED CAN BE SO DIFFICULT, UNTIL THEY HAVE TRIED IT."

Aufbauend auf den Arbeiten von COOK [16] und KARP [34] gaben LENSTRA/RINNOOY KAN/BRUCKER [41]eine theoretische Begründung dafür, warum die meisten Scheduling-Probleme soviel Schwierigkeiten bei der numerischen Lösung bereiten. Sie wiesen für gewisse Scheduling-Probleme nach, daß sie zur Klasse der $\mathcal{NP}$-vollständigen Probleme gehören. An dieser Stelle wird ein Satz von LENSTRA/RINNOOY KAN/BRUCKER wiedergegeben.

Satz 2.2:
(Vgl. z.B. LENSTRA/RINNOOY KAN/BRUCKER [41, S. 354])
Folgendes Scheduling-Problem gehört zur Klasse der $\mathcal{NP}$-vollständigen Probleme: Gegeben seien N Aufträge, die in unterschiedlichen technologischen Reihenfolgen auf zwei Maschinen so zu bearbeiten sind, daß die Zykluszeit minimal wird.

Beweisidee: Es wird eine in der Zeitkomplexität polynomiell begrenzte Transformation des KNAPSACK-Problems auf das oben beschriebene Scheduling-Problem vorgenommen.

Da man sich mit der numerischen Lösung von Ablaufplanungsmodellen etwa seit 1954 in der Literatur sehr stark beschäftigte, erkannte man die begrenzte Einsatzfähigkeit von Algorithmen, die die exakte und damit minimale Lösung ermittelten. Es wurden Verfahren kon-

struiert, die nicht unbedingt die exakte Lösung berechnen, sondern zumindest Näherungslösungen konstruieren können. Bei der Entwicklung solcher Algorithmen wurde vor allen Dingen auf geringe Rechenzeiten und auf die Güte der Näherungslösungen geachtet.

Aus den vorhergehenden Ausführungen dürfte klar sein, daß solche Algorithmen in polynomieller Zeitkomplexität arbeiten und gleichzeitig die ermittelten Zielwerte nur geringfügig von der optimalen Lösungsalternative abweichen sollten. Diese beiden Forderungen an Rechenverfahren sind im allgemeinen konträre Ziele.

Ein Algorithmus, der nur eine geringe Rechenzeit erfordert, wird Zielwerte liefern, die relativ weit von der optimalen Lösung entfernt sind, oder über deren Güte keine Aussage gemacht werden kann.

Im folgenden Kapitel wird versucht, die Zeitkomplexität von Algorithmen zur Erzeugung zulässiger Ablaufpläne, die auch als Näherungsverfahren bezeichnet werden können, abzuschätzen.

## 3. Algorithmen zur Erzeugung zulässiger Ablaufpläne

### 3.1. Zulässige Ablaufpläne

MARIMONT beschreibt in [45] ein graphentheoretisches Verfahren, um zu überprüfen, ob ein Ablaufplan zyklenfrei, d.h. zulässig ist. Dargestellt sei ein Graph F, der die technologischen Reihenfolgebedingungen und die Auftragsfolgen der einzelnen Maschinen wiedergebe. F besteht aus der Knotenmenge V - der Menge aller durchzuführenden Bearbeitungen - und einer Pfeilmenge $C \cup D^*$, die die technologischen Bedingungen und die Auftragsfolgebedingungen enthält. Mit Hilfe des Graphen F kann dann der Begriff der Adjazenz-(Präzedenz-) Matrix definiert werden.

Definition 3.1:
Die einem Ablaufgraphen $F = \{V, C \cup D^*, P\}$ zugeordnete $(N*M, N*M)$-Matrix A(F) mit den Elementen

$$a_{IJ,KL} := \begin{cases} 1 & \text{wenn } (IJ,KL) \in C \cup D^* \\ 0 & \text{sonst} \end{cases}$$

heißt Adjazenz- (Präzedenz-) Matrix von F. Die Elemente $a_{IJ,KL}$ sollen dabei wie folgt angeordnet sein: IJ gebe den Zeilenindex mit I = 1,...,N; J = 1,...,M an. KL bezeichne den Spaltenindex mit K = 1,...,N; L = 1,...,M;

Beispiel 3.1:
Für Beispiel 1.2 sei folgender Graph F gegeben.

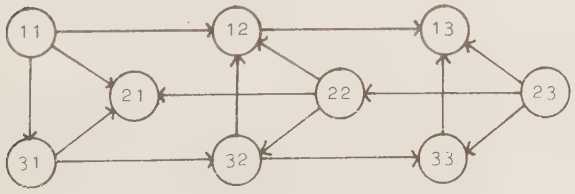



Abbildung 3

Dann ist dem Graphen F, der auch als Ablaufgraph F bezeichnet wird, folgende Matrix A(F) zugeordnet:

| | 11 | 12 | 13 | 21 | 22 | 23 | 31 | 32 | 33 |
|---|---|---|---|---|---|---|---|---|---|
| 11 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| 12 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 13 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 21 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 22 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 23 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 |
| 31 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 32 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 33 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

Die Spaltensummen der Matrix A(F) geben die Anzahl der Pfeile an, die in den betreffenden Knoten einmünden, während die Zeilensummen angeben, wieviel Pfeile vom betreffenden Knoten weggehen. Ist nun entweder eine Zeilensumme oder Spaltensumme Null, so kann der Knoten aus der Adjazenzmatrix entfernt werden, da sich der betreffende Knoten eindeutig anordnen läßt.

Algorithmus 1:

*Name:* Verfahren von MARIMONT [45] zur Ermittlung der Zulässigkeit eines Ablaufplans.

*Voraussetzung:* Gegeben sei die (N*M,N*M)-Adjazenzmatrix A(F).

*Initialisierung:* 1. Ermittle die Zeilen- und Spaltensummen der Matrix A(F).

*Iteration:* 2. Für jede Nullzeilensumme IJ subtrahiere die Spalte IJ von der Spalte der Zeilensummen und entferne aus A(F) Zeile und Spalte IJ.

3. Für jede Nullspaltensumme KL subtrahiere die Zeile KL von der Zeile der Spaltensummen und entferne Zeile und Spalte KL.

*STOPKRITERIUM:* Enthält A(F) keine Spalten und Zeilen mehr? Wenn ja, dann STOP; F stellt einen zulässigen Ablaufplan dar. Wenn nein, dann prüfe, ob in der reduzierten Matrix $A_k(F)$ in der Zeile der Spaltensummen oder in der Spalte der Zeilensummen mindestens ein Element Null ist?
Wenn ja, wiederhole 2. und 3.
Wenn nein, dann STOP. F enthält einen Zyklus.

Bemerkung 3.1:
Schritt 1 benötigt $2*N^2*M^2$ Additionen; Schritte 2 und 3 werden jeweils maximal N*M-mal durchgeführt und benötigen jeweils maximal N*M Subtraktionen. Algorithmus 1 ermittelt in $O(N^2*M^2)$ elementaren Operationen die Zulässigkeit oder Nichtzulässigkeit eines Ablaufgraphen F.

Algorithmus 1 soll an Beispiel 3.1 demonstriert werden.

$A_o(F)$ =

| | 11 | 12 | 13 | 21 | 22 | 23 | 31 | 32 | 33 | |
|---|---|---|---|---|---|---|---|---|---|---|
| 11 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 3 |
| 12 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 13 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 21 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 22 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 3 |
| 23 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 3 |
| 31 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 2 |
| 32 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 2 |
| 33 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| | 0 | 3 | 3 | 3 | 1 | 0 | 1 | 2 | 2 | |

Nach Streichen von Zeile und Spalte 3, Zeile und Spalte 4. und Zeile und Spalte 1 und 6 erhält man die reduzierte Matrix $A_2(F)$.

$A_1(F)$ =

| | 12 | 22 | 31 | 32 | 33 | |
|---|---|---|---|---|---|---|
| 12 | 0 | 0 | 0 | 0 | 0 | 0 |
| 22 | 1 | 0 | 0 | 1 | 0 | 2 |
| 31 | 0 | 0 | 0 | 1 | 0 | 1 |
| 32 | 1 | 0 | 0 | 0 | 1 | 2 |
| 33 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 2 | 0 | 0 | 2 | 1 | |

$A_2(F)$ =

| | 22 | 32 | |
|---|---|---|---|
| 22 | 0 | 1 | 1 |
| 32 | 0 | 0 | 0 |
| | 0 | 1 | |

$A_3(F) = \{\emptyset\}$

Der Graph F ist also ein zyklenfreier Graph, d.h. ein zulässiger Ablaufgraph.

Bemerkung 3.2:

Es wird sich später als wichtig erweisen, alle disjunkten Zyklen (also Zyklen, die keine gemeinsamen Pfeile besitzen) zu kennen. Es soll ein Algorithmus angegeben werden, der alle disjunkten Zyklen in einem Graphen F ermittelt.

Angenommen, man gelangt irgendwann zu einer Matrix $A_k(F)$, in der kein Element der Zeilen- oder Spaltensummen Null ist, dann können alle disjunkten Zyklen mit Algorithmus 2 ermittelt werden.

Algorithmus 2:

*Name:* Verfahren zur Bestimmung aller disjunkter Zyklen in einem Graphen F (vgl. NEUMANN [49], NOLTEMEIER [50]).

*Voraussetzung:* Gegeben sei eine nicht mehr reduzierbare Teilmatrix $A_k(F)$ der Adjazenzmatrix A(F).

*Initialisierung:* Wähle aus $A_k(F)$ eine beliebige Zeile IJ aus und suche ein Element $a_{IJ,KL} = 1$.

*Iteration:* 1. Ersetze $a_{IJ,KL}$ durch Null. Suche in Zeile KL ein Element $a_{KL,RT} = 1$. Setze $a_{KL,RT}$ gleich Null.
2. Wiederhole 1. solange, bis IJ = RT gilt. Ein Zyklus ist gefunden.
3. Streiche aus $A_k(F)$ alle Zeilen und Spalten, die zu Nullzeilen- oder Nullspaltensummen korrespondieren.

*STOPKRITERIUM:* Falls $A_k(F)$ vollkommen reduziert ist, dann STOP. Andernfalls wiederhole 1., 2. und 3.

Bemerkung 3.3:

Wird angenommen, daß $A_k(F)$ gleich $A_o(F)$ ist, so benötigt der Algorithmus 2 $O(N^2*M^2)$ elementare Operationen.

Bemerkung 3.4:

Wie am Beispiel 3.1 ersichtlich ist, ist die Matrix A(F) „dünn" besetzt, d.h., sehr viele Elemente sind Null. Bezeichnet man mit dem Besetzungsgrad den prozentualen Anteil der von Null verschiedenen Elemente, so gilt

$$\text{Besetzungsgrad} := \frac{\text{Summe der Zeilen(Spalten)summen}}{N^2*M^2} * 100.$$

Bei Beispiel 3.1 ist die Matrix nur zu 18,5% besetzt. Als Abschätzung des Zählers des vorstehenden Bruches läßt sich folgendes angeben: Der Zähler besitzt bei N Aufträgen und M Maschinen den Wert N*(M-1) + N*(N-1)M/2, wobei N*(M-1) die Zahl der technologischen Reihenfolgen angibt und N*(N-1)*M/2 die Zahl der Auftragsfolgebedingungen angibt. Also gilt für den Besetzungsgrad der Adjazenzmatrix A(F) eines Ablaufgraphen F

$$\text{Besetzungsgrad} = \frac{N*(M-1)+N*(N-1)*M/2}{N^2*M^2} = \frac{(M-1)+(N-1)*M/2}{N*M^2}$$

$$= \frac{(1-\frac{1}{M})+(N-1)/2}{N*M}$$

Für große Werte von M geht der Ausdruck $\frac{(1-\frac{1}{M})+(N-1)/2}{N*M}$ über in $\frac{1+(N-1)/2}{N*M}$. Beispielsweise gilt für N = 10 und M = 10, daß der Besetzungsgrad der Matrix A(F) ≈ 5,5% ist. Der genaue Besetzungsgrad beträgt 5,4%. Würde man solche Matrizen wie üblich in zweidimensionalen Feldern speichern, so würde dies sehr viel Speicherplatz benötigen. So würde man für ein Beispiel mit 10 Aufträgen und 10 Maschinen 10.000 Speicherplätze benötigen. Die spezielle Struktur von A(F) - die Elemente sind Null oder Eins - läßt jedoch eine bessere Speicherung zu. A(F) läßt sich „bitweise" abspeichern. Angenommen, ein Speicherwort (wie z.B. die TR 440 am Rechenzentrum der Universität des Saarlandes) verfüge über 52 Bits, von denen 48 Bits zur Speicherung zur Verfügung stehen. Soll die Adjazenzmatrix A(F) „bitweise" gespeichert werden, so genügen maximal (ENTIER ((N*M)/48 + 1)*N*M) Speicherworte. Für N = 10 und M = 10 würden etwa 300 Speicherworte genügen.

Falls eine „bitweise" Speicherung der Adjazenzmatrix A(F) vorgenommen wird, müssen schnelle Unterprogramme in Assembler zur Verfügung stehen, um arithmetische Operationen in der Matrix A(F) durchzuführen. Wird allerdings angestrebt, ein Verfahren zur Überprüfung der Zulässigkeit von Ablaufplänen in einer höheren Programmiersprache, wie z.B. ALGOL oder FORTRAN, zu kodieren, so kann eine „bitweise" Speicherung nicht empfohlen werden. Es empfiehlt sich z.B., auf die FOUR-ROW-Technik für „dünn" besetzte Matrizen zurückzugreifen (vgl. z.B.: BERTZISS [10], KNUTH [35]. Dabei dienen die ersten drei

Zeilen dazu, den Zeilenindex, den Spaltenindex und den Wert des von Null verschiedenen Koeffizienten aufzunehmen. In der vierten Zeile wird eine Verkettung der Spalten in der Weise vorgenommen, daß das Element in der vierten Zeile Aufschluß darüber gibt, wo das nächste von Null verschiedene Element der betreffenden Spalte in der FOUR-ROW-Matrix ist.

Da in der Adjazenzmatrix nur Nullen und Einsen vorkommen und nur von Null verschiedene Elemente gespeichert werden sollen, braucht der Wert des Koeffizienten nicht gespeichert zu werden. Die FOUR-ROW-Technik kann in eine THREE-ROW-Technik modifiziert werden.

Die Adjazenzmatrix A(F) von Beispiel 3.1 könnte dann wie folgt gespeichert werden:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Zeile 1 | : 1 | 1 | 1 | 2 | 5 | 5 | 5 | 6 | 6 | 6 | 7 | 7 | 8 | 8 | 9 |
| Zeile 2 | : 2 | 4 | 7 | 3 | 2 | 4 | 8 | 3 | 5 | 9 | 4 | 8 | 2 | 9 | 3 |
| Zeile 3 | : 5 | 6 | 0 | 10 | 13 | 11 | 12 | 15 | 0 | 14 | 0 | 0 | 0 | 0 | 0 |

Die Anzahl der benötigten Speicherworte beträgt bei einer THREE-ROW-Darstellung $3*(N*(M-1) + N*(N-1)*M/2)$.
Beispielsweise würden für N = 10 und M = 10 1.620 Speicherworte benötigt.
Neben der erwähnten Speicherplatzersparnis kann auch bei großen Adjazenzmatrizen Rechenzeit dadurch gespart werden, daß nur arithmetische Operationen mit Elementen ausgeführt werden, die den Wert Eins besitzen.

In Algorithmus 1 werden bei der Initialisierung $2*N^2*M^2$ Additionen bei üblicher Speicherung benötigt. Wird die Adjazenzmatrix in einer THREE-ROW-Technik gespeichert, so sind das nur insgesamt $2*(N*(M-1) + N*(N-1)*M/2)$ Additionen.

Für N = 10 und M = 10 würden bei üblicher Speichertechnik 20.000 Additionen, bei einer Speicherung in THREE-ROW-Technik 1.080 Additionen benötigt.
Damit dürfte die Wichtigkeit der verwendeten Datenstrukturen (hier Speicherung von dünn besetzten Matrizen) angedeutet worden sein.

## 3.2. Semiaktive Ablaufpläne

Die Anzahl der aus einem disjunktiven Graphen G = (V,C,D,P) resultierenden Graphen F = (V,C ∪ D*,P) ist durch die Zahl $2^{N*(N-1)*M/2}$ nach oben beschränkt. Diese obere Schranke ist allerdings sehr grob für die mögliche Anzahl aller zulässigen Ablaufgraphen, da sehr viele Graphen einen Zyklus enthalten und damit unzulässig sind. Einem zulässigen Ablaufgraphen F können in einfacher Weise unendlich viele zulässige Ablaufpläne H ∈ X derart zugeordnet werden, indem die Beginnzeitpunkte $h_{IJ}$ beliebig verschoben werden unter Beachtung der technologischen Reihenfolgen und der Maschinenreihenfolgen. Um einem Ablaufgraphen F, der eine vollkommen reduzierbare Adjazenzmatrix A(F) besitzen möge, nur einen zulässigen Ablaufplan H ∈ X zuordnen zu können, wurde das Konzept der *semiaktiven* Ablaufpläne eingeführt.

**Definition 3.2:**
(Vgl. DINKELBACH [18], CONWAY/MAXWELL/MILLER [15])
Ein zu einem zulässigen Ablaufgraphen F korrespondierender Ablaufplan H(F) ∈ X heißt semiaktiv bezüglich X, wenn kein anderer Ablaufplan H'(F) ∈ X existiert, bei dem alle Beginnzeitpunkte $h'_{IJ}$ kleiner oder gleich den entsprechenden Beginnzeitpunkten beim Ablaufplan H(F) sind und wenigstens ein Beginnzeitpunkt $h'_{IJ}$ echt kleiner als der entsprechende Beginnzeitpunkt $h_{IJ}$ beim Ablaufplan H ist. Oder kürzer:
H(F) heißt semiaktiv genau dann, wenn ∄ H'(F) mit H'(F) ≦ H(F).

Bemerkung 3.5:

Beim semiaktiven Ablaufplan H(F) sind die Reihenfolgen der Aufträge auf den Maschinen und die technologischen Reihenfolgen durch den Graphen F vorgeschrieben. Im allgemeinen sind jedoch die Auftragsfolgen auf den M Maschinen zu bestimmen. Dies kann mit Algorithmus 3 geschehen. Um diesen Algorithmus darstellen zu können, müssen einige Bezeichnungen definiert werden.

Definition 3.3:

| Bezeichnung | Bedeutung | Bemerkungen |
|---|---|---|
| *NM* | Indexmenge der N*M Bearbeitungen | *NM* enthalte die Bearbeitungen in lexikographischer Reihenfolge (d.h., die Operationen IJ sind nach Maschinen gruppiert). |
| *CNM* | Indexmenge der technologischen Nachfolgebeziehungen | Besitzt eine Operation keine Nachfolgebeziehung, so wird dies mit Null gekennzeichnet. |
| *PNM* | Menge der Bearbeitungszeiten für alle Operationen aus der Indexmenge *NM* | *PNM* enthält die Matrix P spaltenweise. |
| *HANM* | Menge der frühesten Verfügbarkeitstermine der Operationen aus *NM* | Die Größen in *HANM* sind auftragsbezogene früheste Starttermine. Eine Operation kann dann erst bearbeitet werden, wenn die technologische Vorgängeroperation abgeschlossen ist. Besitzt eine Operation keine Vorgängeroperation, so ist sie zum Zeitpunkt Null bereit, bearbeitet zu werden. |

| | | |
|---|---|---|
| *HMNM* | Menge der frühesten Starttermine bezogen auf die Maschinen 1,...,M | Die Größen in *HMNM* sind maschinenbezogene früheste Starttermine. Eine Operation kann erst dann auf Maschine J bearbeitet werden, wenn die Bearbeitung, die einem Auftrag auf dieser Maschine vorangeht, abgeschlossen ist. |
| *FNM* | Nummern der Reihenfolgen der Operationen pro Maschine | *FNM* enthält M Permutationen der Zahlen 1,...,N. |
| *HNM* | Menge der Starttermine der Operationen aus der Indexmenge *NM* | Der Starttermin einer Operation IJ ergibt sich als das jeweilige Maximum der beiden Größen: frühester (auftragsbezogener) Starttermin und Fertigstellungstermin der Operation, die IJ auf Maschine J vorangeht. |
| *DNM* | Menge der Fertigstellungszeitpunkte aller Operationen aus der Indexmenge *NM* | Die Fertigstellungszeitpunkte ergeben sich als Summe aus Startzeitpunkt $h_{IJ}$ und Bearbeitungszeit $p_{IJ}$. |

Algorithmus 3:

(Vgl. ASHOUR [3], CONWAY/MAXWELL/MILLER [15], SEELBACH [54])

*Name:* Verfahren zur Erzeugung *semiaktiver* Ablaufpläne.

*Voraussetzung:* Gegeben sei G = (V,C,D,P).

*Initialisierung:* Initialisiere die Mengen *NM*, *CNM*, *PNM*. Kennzeichne in *HANM* die jeweils erste Bearbeitung eines Auftrags mit einer Null. Fasse alle ersten Bearbeitungen zu einer Menge *SNM* zusammen. Setze alle Elemente aus *FNM*, *HNM*, *DNM* gleich Null.

*Iteration:*

1. Wähle aus *SNM* einen Index IJ aus.
2. Setze $FNM(IJ) = \max_{K=1,\ldots,N} \{FNM(KJ)\} + 1.$
3. Setze $DNM(IJ) = \max \{HANM(IJ), \max_{K=1,\ldots,N} \{DNM(KJ)\}\} + PNM(IJ).$
4. Entferne aus *SNM* den Index IJ und ergänze *SNM* um den technologischen Nachfolger aus *CNM*. Dies sei Index IK, $K \in \{1,\ldots,M\}$; $K \neq J$. Besitzt IJ keinen Nachfolger mehr, so verringere die Mächtigkeit der Menge *SNM* um 1.
5. Setze $HANM(IK) = DNM(IJ)$.

*STOPKRITERIUM:* Ist $SNM = \emptyset$? Wenn ja, dann STOP. Ein semiaktiver Plan wurde gefunden. Wenn nein, wiederhole 1. bis 5.

**Beispiel 3.2:**

Für Beispiel 1.2 soll ein semiaktiver Ablaufplan konstruiert werden.

| *NM* | *CNM* | *PNM* | *HANM* | *FNM* | *HNM* | *DNM* |
|---|---|---|---|---|---|---|
| 11 | 12 | 4 | 0 | 1 | 0 | 4 |
| 21 | 0 | 3 | 19 | 2 | 19 | 22 |
| 31 | 32 | 2 | 0 | 3 | 22 | 24 |
| 12 | 13 | 2 | 4 | 1 | 4 | 6 |
| 22 | 21 | 7 | 12 | 2 | 12 | 19 |
| 32 | 33 | 4 | 24 | 3 | 24 | 28 |
| 13 | 0 | 1 | 6 | 1 | 6 | 7 |
| 23 | 22 | 5 | 0 | 2 | 7 | 12 |
| 33 | 0 | 6 | 28 | 3 | 28 | 34 |

Bemerkung 3.6:
Für die Darstellung von Ablaufplänen haben sich *Ganttdiagramme* durchgesetzt. Der obige semiaktive Ablaufplan sieht dann folgendermaßen aus:

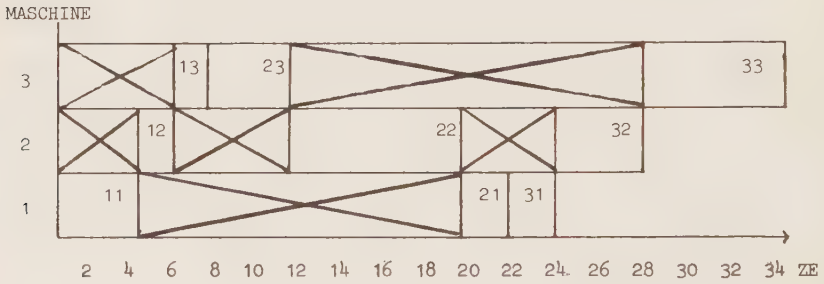



Abbildung 4

Bemerkung 3.7:
Aus der letzten Spalte der Tabelle von Beispiel 3.2 lassen sich die Zielwerte der in Kapitel 1.2 diskutierten Zielfunktionen ablesen.

Zykluszeit Z := 34 ZE,
Durchlaufzeit GD := 7 + 22 + 34 = 63 ZE,
Belegungszeit GB := 24 + 28 + 34 = 86 ZE.

Unterschiedliche semiaktive Ablaufpläne können durch die Auswahl in Schritt 1 erzeugt werden.

Satz 3.1:
Algorithmus 3 erzeugt in $O(N^2*M)$ Schritten einen semiaktiven Ablaufplan.

Beweis: Zur Feststellung des Rechenaufwandes wird der größte „Engpaß" gesucht: Schritt 1 wird N*M-mal durchlaufen. In Schritt 2 wird das Maximum einer Menge mit N Elementen gesucht. Dies kann in O(N) Schritten geschehen (vgl.

Kapitel 2). Insgesamt wird diese Maximumbildung N*M-mal ausgeführt. In Schritt 3 wird zunächst das Maximum einer M elementigen Menge gesucht und dann das Maximum einer zweielementigen Menge gebildet. Insgesamt beträgt der Rechenaufwand also O(N). Schritt 3 wird N*M-mal durchlaufen. Daraus ergibt sich ein Rechenaufwand von $O(N^2*M)$.

## 3.3. Aktive Ablaufpläne

Bei der Konstruktion von semiaktiven Ablaufplänen werden Operationen zufällig aus der Menge *SNM* der verfügbaren Operationen ausgewählt. Dadurch kann es geschehen, daß eine Operation hinausgezögert wird. Es kann zu großen Leerzeiten von einzelnen Maschinen kommen. Beispielsweise ist die Operation 23 aus Abbildung 4 zum Zeitpunkt Null verfügbar. Aufgrund der zufälligen Auswahlregel beginnt die Bearbeitung von Operation 23 aber erst zum Zeitpunkt 7. Die Maschine 3 ist aber vom Zeitpunkt 0 bis zum Zeitpunkt 6 überhaupt nicht belegt. Würde die Bearbeitung von Operation 23 vor Operation 13 gelegt, so könnten unter Umständen Nachfolgeoperationen früher beginnen, was eine Verringerung von zu minimierenden Zielkriterien nach sich ziehen könnte. Operation 23 benötigt 5 ZE zur Bearbeitung, d.h., Operation 13 könnte wie bisher zum Zeitpunkt 6 beginnen.

Bei der Definition von semiaktiven Ablaufplänen wird einschränkend von gleichen Auftragsfolgen bei den zu vergleichenden Ablaufplänen für jede Maschine ausgegangen. Wie oben ausgeführt wurde, kann dies aber zu unnötigen Leerzeiten führen. Gesucht sind semiaktive Ablaufpläne, bei denen keine so großen Leerzeiten auftreten, in denen eine verfügbare Operation vollkommen bearbeitet werden kann und keine der übrigen Operationen verzögert wird. Solche semiaktiven Ablaufpläne sollen aktiv heißen.

Eine Möglichkeit zur Konstruktion von aktiven Plänen wäre es, einen semiaktiven Ablaufplan zu erzeugen und sukzessiv die Leerzeitenintervalle der Maschinen zu kontrollieren, ob nicht eine Bearbeitung vorgezogen werden kann. Diese Vorgehensweise ist recht mühsam. Es soll für Algorithmus 3 eine Modifikation angegeben werden, damit sofort aktive Pläne erzeugt werden. Zuerst soll jedoch eine exakte Definition des Begriffs des aktiven Ablaufplans gegeben werden.

Definition 3.3:
(Vgl. DINKELBACH [18], GIFFLER/THOMPSON [24])
Ein Ablaufplan $H \in X$ heißt aktiv bezüglich X, wenn kein anderer Ablaufplan $H' \in X$ existiert, bei dem alle Starttermine $h'_{IJ}$ kleiner oder gleich den entsprechenden Startterminen beim Ablaufplan H sind und wenigstens ein Starttermin $h'_{I_rJ}$ echt kleiner als der entsprechende Beginnzeitpunkt beim Ablaufplan H ist. Oder kürzer:
$H \in X$ ist aktiv bezüglich X genau dann: $\nexists\, H' \in X$ mit $H' \leqq H$.

Algorithmus 4:
(Vgl. GIFFLER/THOMPSON [24], RINNOOY KAN [51], SEELBACH [54])

*Name:* Verfahren zur Erzeugung *aktiver* Ablaufpläne.

*Voraussetzung:* Gegeben sei G = (V,C,D,P).

*Initialisierung:* Initialisiere die Mengen *NM*, *CNM*, *PNM*. Kennzeichne in *HANM* die jeweils erste Bearbeitung eines Auftrags mit einer Null. Fasse alle ersten Bearbeitungen zu einer Menge *SNM*. Setze alle N*M Elemente aus *HMNM*, *FNM*, *HNM*, *DNM* gleich Null.

*Iteration:*

1. Bestimme für alle Indizes IJ $\in$ *SNM* die Größe $HNM(IJ) = \max\{HANM(IJ), HMNM(IJ)\}$ mit $HMNM(IJ) = \max_{K=1,\ldots,N}\{HMNM(KJ)\}$.
2. Bestimme für alle Indizes IJ $\in$ *SNM* die Fertigstellungstermine $DNM(IJ) = HNM(IJ) + PNM(IJ)$.
3. Bestimme für alle Maschinen J, die in der Indexmenge SNM vorkommen, die Größe $DMIN(J) = \min_{IJ \in SNM}\{DNM(IJ)\}$.
4. Bestimme unter allen Größen $DMIN(J)$ das Minimum.
5. Fasse alle Operationen IJ mit $HNM(IJ) < \min_{J} DMIN(J)$ zu einer Indexmenge *SNM1* zusammen.
6. Wähle aus *SNM1* einen Index IJ aus.
7. Setze $FNM(IJ) = \max_{K=1,\ldots,N}\{FNM(KJ)\} + 1$.
8. Setze $DNM(IJ) = HNM(IJ) + PNM(IJ)$.
9. Entferne aus *SNM* den Index IJ und ergänze *SNM* um den technologischen Nachfolger aus *CNM*. Dies sei Index IK $K \in \{1,\ldots,M\}$; $K \neq J$. Besitzt IJ keinen Nachfolger mehr, so verringere sich die Mächtigkeit der Menge um 1.
10. Setze $HANM(IK) = DNM(IJ)$ und $HMNM(IJ) = DNM(IJ)$.

*STOPKRITERIUM:* Ist *SNM* = Ø? Wenn ja, dann STOP. Ein aktiver Ablaufplan wurde gefunden. Wenn nein, wiederhole 1. bis 10.

Beispiel 3.3:

Für Beispiel 1.2 soll ein aktiver Ablaufplan erzeugt werden.

| NM | CNM | PNM | HANM | HMNM | FNM | HNM | DNM |
|---|---|---|---|---|---|---|---|
| 11 | 12 | 4 | 0 | 4 | 1 | 0 | 4 |
| 21 | 0 | 3 | 12 | 15 | 3 | 12 | 15 |
| 31 | 22 | 2 | 0 | 6 | 2 | 4 | 6 |
| 12 | 13 | 2 | 4 | 18 | 3 | 16 | 18 |
| 22 | 21 | 7 | 5 | 12 | 1 | 5 | 12 |
| 32 | 33 | 4 | 6 | 16 | 2 | 12 | 16 |
| 13 | 0 | 1 | 18 | 23 | 3 | 22 | 23 |
| 23 | 22 | 5 | 0 | 5 | 1 | 0 | 5 |
| 33 | 0 | 6 | 16 | 22 | 2 | 16 | 22 |

Graphische Darstellung:

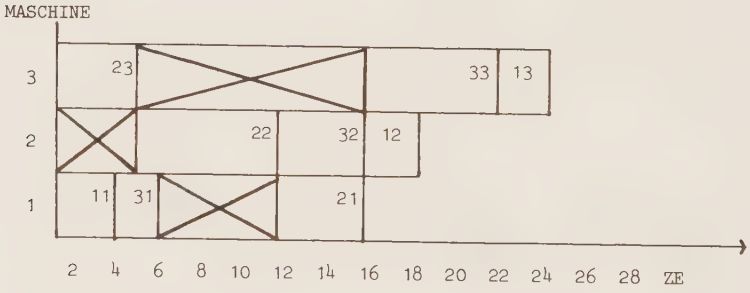



Abbildung 5

Bemerkung 3.8:

Aus der letzten Spalte der Tabelle von Beispiel 3.3 lassen sich die Werte der drei Zielkriterien Zykluszeit, Durchlaufzeit und Belegungszeit errechnen.

Zykluszeit Z := 23 ZE,
Durchlaufzeit GD := 23 + 15 + 22 = 60 ZE,
Belegungszeit BG := 15 + 18 + 23 = 56 ZE.

Unterschiedliche aktive Ablaufpläne lassen sich durch unterschiedliche Auswahl in Schritt 6. erreichen.

Algorithmus 4 unterscheidet sich von Algorithmus 3 dadurch, daß nur solche Bearbeitungen ausgewählt werden können, die einen Starttermin aufweisen, der kleiner als der Endtermin aller anderen verfügbaren Operationen zuzüglich ihrer Bearbeitungszeit ist. Dies ist genau das Kriterium, das aktive Ablaufpläne von semiaktiven Ablaufplänen unterscheidet. Satz 3.2 gibt Auskunft darüber, wie die Zusammenhänge zwischen semiaktiven und aktiven Ablaufplänen sind.

Satz 3.2:
Jeder aktive Ablaufplan $H \in X$ ist auch semiaktiv bezüglich X (vgl. z.B. BAKER [4], SEELBACH [54]).

Beweis: $H \in X$ sei aktiv. Zu zeigen ist: $\nexists\ H'(F) \in X$ mit $H'(F) \leqq H(F)$. H induziert einen Ablaufgraphen F. Dann folgt sofort aus der Definition eines aktiven Ablaufplans: $\nexists\ H'(F) \in X$, so daß $H'(F) \leqq H(F)$.

Die Umkehrung dieses Satzes gilt nicht im allgemeinen. Beispielsweise ist der folgende graphisch dargestellte Ablaufplan $H_2$ semiaktiv, aber nicht aktiv.

Beispiel 3.4:

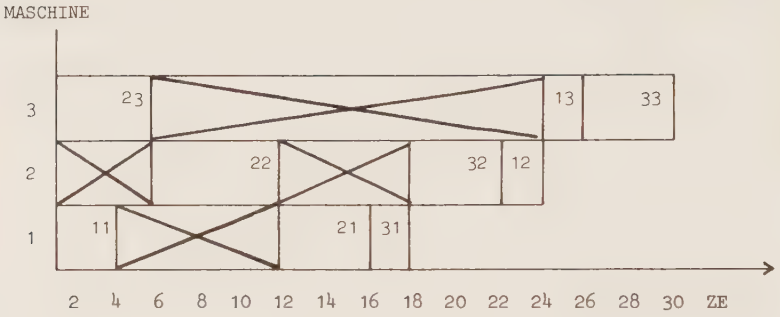



Abbildung 6

Der in Abbildung 6 dargestellte Ablaufplan lautet in Matrizenschreibweise:

$$H = \begin{pmatrix} 0 & 21 & 23 \\ 12 & 5 & 0 \\ 15 & 17 & 24 \end{pmatrix}.$$

Der Ablaufplan H' aus Beispiel 3.3 lautet:

$$H' = \begin{pmatrix} 0 & 16 & 22 \\ 12 & 5 & 0 \\ 4 & 12 & 16 \end{pmatrix}.$$

Es gilt: $H' \leq H$.

Satz 3.3 gibt über den Rechenaufwand von Algorithmus 4 Aufschluß.

Satz 3.3:

Algorithmus 4 erzeugt in $O(N^3 * M^2)$ Schritten einen aktiven Ablaufplan.

Beweis: Schritt 1 benötigt zur Bestimmung von HMNM(IJ) O(N) Schritte. Maximal enthält die Menge *SNM* N Elemente. Daraus ergibt sich ein Rechenaufwand von $O(N^2)$ für Schritt 1.

Schritt 2 hat die Zeitkomplexität O(N). In Schritt 3 wird im schlechtesten Fall für M Maschinen jeweils das Minimum einer N elementigen Menge bestimmt. Die Zeitkomplexität beträgt also O(N*M). Schritt 4 benötigt O(M). Schritt 5 kann in O(N) abgewickelt werden. Die Schritte 6, 7, 8, 9 und 10 haben die gleiche Zeitkomplexität wie in Algorithmus 3. Insgesamt werden die Schritte 1 bis 10 insgesamt N*M-mal durchlaufen. Daraus ergibt sich für die Schritte 1 und 3 eine Zeitkomplexität von $O(N^3*M+N^2*M^2) = O(N^3*M^2)$.

## 3.4. Unverzögerte Ablaufpläne

Bei der Erzeugung aktiver Ablaufpläne lassen sich inaktive Ablaufpläne ausschließen, indem Leerzeitenintervalle vermieden werden, in denen eine Bearbeitung vollkommen durchgeführt werden könnte. Unverzögerte Ablaufpläne sind dadurch gekennzeichnet, daß jede Maschine zum frühestmöglichen Zeitpunkt mit einer verfügbaren Operation belegt wird.

Zu jedem zulässigen Ablaufplan H, bei dem die Startzeitpunkte $h_{IJ}$ und damit die Auftragsfolgen für alle Maschinen festgelegt sind, korrespondiert eine Leerzeitenintervallmatrix.

**Definition 3.4:**
Eine zu einem zulässigen Ablaufplan H korrespondierende (M,N)-Matrix heißt Leerzeitenintervallmatrix LE, wenn jede Zeile I (I = 1,...,M) die Leerzeitenintervalle der Maschine J (J $\in$ {1,...,N}) zwischen zwei aufeinanderfolgenden Bearbeitungen wiedergibt. Die erste Spalte der Matrix LE enthält das Leerzeitenintervall, das zwischen dem Zeitpunkt 0 und dem Starttermin des Auftrags vergeht, der an der ersten Stelle der Auftragsfolge einer Maschine steht.

Beispiel 3.5:

In Beispiel 3.3 wurde der folgende Ablaufplan erzeugt:

$$H = \begin{pmatrix} 0 & 16 & 22 \\ 12 & 5 & 0 \\ 4 & 12 & 16 \end{pmatrix}.$$

Die Bearbeitungszeitenmatrix P lautet:

$$P = \begin{pmatrix} 4 & 2 & 1 \\ 3 & 7 & 5 \\ 2 & 4 & 6 \end{pmatrix}.$$

Die Leerzeitenintervallmatrix LE kann dann konstruiert werden:

$$LE = \begin{pmatrix} [0:0] & [0:0\ ] & [6:12] \\ [0:5] & [0:0\ ] & [0:0\ ] \\ [0:0] & [5:16] & [0:0\ ] \end{pmatrix}.$$

Definition 3.5:

Ein aktiver Plan $H \in X$ heißt unverzögert, wenn kein Freigabetermin *HANM*(IJ) (I = 1,...,N; J = 1,...,M) *echt* aus einem Leerzeitenintervall der Zeile I der Matrix LE ist.

Der folgende Algorithmus konstruiert unverzögerte Pläne.

Algorithmus 5:

(Vgl. ASHOUR [3], CONWAY/MAXWELL/MILLER [15], HELLER/LOGEMANN [28], SEELBACH [54])

*Name:* Verfahren zur Erzeugung *unverzögerter* Ablaufpläne.

*Voraussetzung:* Gegeben sei G = (V,C,D,P).

*Initialisierung:* Initialisiere die Mengen *NM*, *CNM* und *PNM*. Kennzeichne in *HANM* die jeweils erste Bearbeitung eines Auftrags mit einer Null. Fasse die Menge aller ersten Bearbeitungen

zu einer Indexmenge *SNM* zusammen. Setze alle N*M Elemente aus *HMNM*, *FNM*, *HNM* und *DNM* gleich Null.

*Iteration:*

1. Bestimme für alle Indizes IJ $\in$ *SNM* die Größe *HNM*(IJ).

$HNM(IJ) = \max\{HANM(IJ), HMNM(IJ)\}$, wobei

$HMNM(IJ) = \max_{K=1,\ldots,N}\{HMNM(KJ)\}$.

2. Bestimme die Größe

$HMIN = \min_{IJ \in SNM}\{HNM(IJ)\}$.

3. Fasse alle Indizes aus *SNM*, für die *HMIN* = *HNM*(IJ) gilt, zu einer Indexmenge *SNM2* zusammen.

4. Wähle aus *SNM2* einen Index IJ aus.

5. Setze $FNM(IJ) = \max_{K=1,\ldots,N}\{FNM(KJ)\} + 1$.

6. Setze *DNM*(IJ) = *HNM*(IJ) + *PNM*(IJ).

7. Entferne aus *SNM* den Index IJ und ergänze *SNM* um den technologischen Nachfolger aus *CNM*. Dies sei Index IK K $\in$ {1,...,M}; K $\neq$ J. Besitzt IJ keinen Nachfolger mehr, so verringere die Mächtigkeit der Menge *SNM* um 1.

8. Setze *HANM*(IK) = *DNM*(IJ) und *HMNM*(IJ) = *DNM*(IJ).

*STOPKRITERIUM:* Ist *SNM* = Ø? Wenn ja, dann STOP. Ein unverzögerter Ablaufplan wurde gefunden. Wenn nein, wiederhole 1. bis 8.

**Beispiel 3.6:**

Für Beispiel 1.2 soll ein unverzögerter Ablaufplan erzeugt werden.

| NM | CNM | PNM | HANM | HMNM | FNM | HNM | DNM |
|---|---|---|---|---|---|---|---|
| 11 | 12 | 4 | 0 | 4 | 1 | 0 | 4 |
| 21 | 0 | 3 | 13 | 16 | 3 | 13 | 16 |
| 31 | 32 | 2 | 0 | 6 | 2 | 4 | 6 |
| 12 | 13 | 2 | 4 | 6 | 1 | 4 | 6 |
| 22 | 21 | 7 | 5 | 13 | 2 | 6 | 13 |
| 32 | 33 | 4 | 6 | 17 | 3 | 13 | 17 |
| 13 | 0 | 1 | 6 | 7 | 2 | 6 | 7 |
| 23 | 22 | 5 | 0 | 5 | 1 | 0 | 5 |
| 33 | 0 | 6 | 17 | 23 | 3 | 17 | 23 |

Graphische Darstellung:

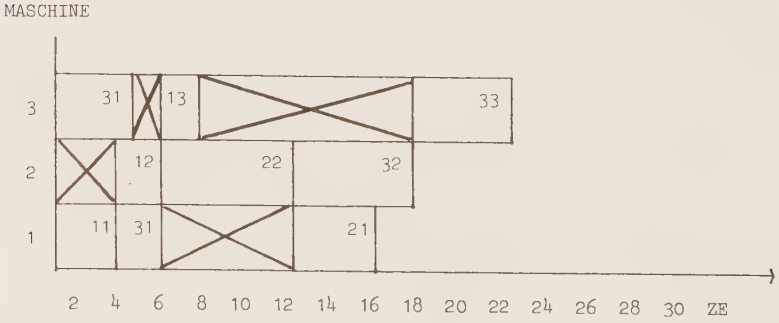



Abbildung 7

Aus der Definition von unverzögerten Ablaufplänen ist ersichtlich, daß jeder unverzögerte Ablaufplan auch aktiv ist. Die Umkehrung dieser Aussage gilt jedoch nicht wie Beispiel 3.7 zeigt.

Beispiel 3.7:

Für Beispiel 1.2 sei der folgende graphisch dargestellte aktive Ablaufplan ermittelt worden:

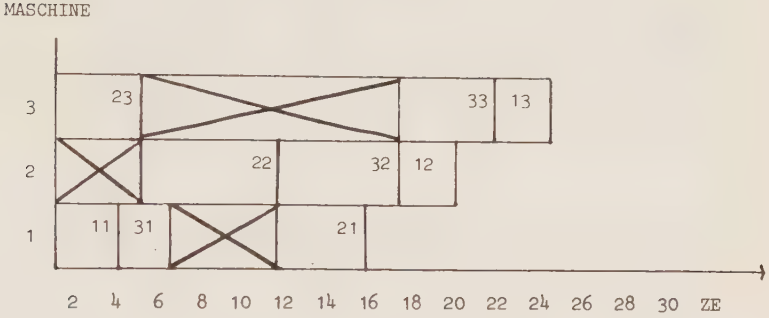



Abbildung 8

Die Leerzeitenmatrix LE lautet dann:

$$LE = \begin{pmatrix} [0:0] & [0:0\ ] & [6:12] \\ [0:5] & [0:0\ ] & [0:0\ ] \\ [0:0] & [5:16] & [0:0\ ] \end{pmatrix}.$$

Der Freigabetermin *HANM*(12) ist der Zeitpunkt 4, d.h. $4 \in [0,5]$. Der in Abbildung 8 dargestellte Ablaufplan ist nicht unverzögert.
Satz 3.4 gibt den Rechenaufwand zur Erzeugung unverzögerter Ablaufpläne an.

Satz 3.4:

Algorithmus 5 erzeugt in $O(N^3*M)$ Schritten einen unverzögerten Ablaufplan.

**Beweis:** Schritt 1 benötigt wie in Algorithmus 4 $O(N^2)$. Schritt 2 besitzt die Zeitkomplexität O(N), da das Minimum einer maximal N elementigen Menge bestimmt werden muß. Auch Schritt 3 kann in O(N) ausgeführt werden. Die Schritte 5, 6, 7 und 8 besitzen die gleiche Zeitkomplexität wie in Algorithmus 3. Die Schritte 1 bis 8 werden N*M-mal ausgeführt.
Also beträgt der Rechenaufwand für den hier formulierten Algorithmus $O(N^3*M)$.

## 3.5. Zusammenhänge zwischen semiaktiven, aktiven und unverzögerten Ablaufplänen

Die Eigenschaften von semiaktiven, aktiven und unverzögerten Ablaufplänen und ihre Beziehungen zueinander sollen in diesem Abschnitt diskutiert werden. Dazu sei die Menge der semiaktiven Ablaufpläne eines Scheduling-Problems mit N Aufträgen und M Maschinen mit SE bezeichnet, die Menge der aktiven Ablaufpläne mit AK, die Menge der unverzögerten Ablaufpläne mit UN und die Menge der optimalen Ablaufpläne bezüglich einer vorgegebenen Zielsetzung mit O bezeichnet.

**Satz 3.5:**

i. $SE \subset X$
ii. $AK \subset SE \subset X$
iii. $UN \subset AK \subset SE \subset X$
iv. SE enthält mindestens einen Plan aus O.
v. AK enthält mindestens einen Plan aus O.
vi. UN enthält nicht notwendigerweise einen Plan aus O.

**Beweis:**

i. Folgt aus Definition 3.2.
ii. Folgt aus Satz 3.2.
iii. Folgt aus Definition 3.5.
iv. Beweis: Angenommen, es existiere ein optimaler Ab-

laufplan H, der nicht in SE sei. Zu H korrespondiert ein Ablaufplan F. Da zu jedem Ablaufgraphen F, der zyklenfreie Ablaufpläne erzeugt, ein semiaktiver Ablaufplan H'(F) gehört, gilt nach Definition 3.2

$$H'(F) \leqq H(F),$$

d.h., die Starttermine $h'_{IJ}$ sind alle kleiner oder gleich den Startterminen $h_{IJ}$ und mindestens ein Starttermin $h'_{IJ}$ ist echt kleiner als der entsprechende Starttermin $h_{IJ}$.

Da H ein optimaler Ablaufplan ist, muß für den Zielwert H' gelten:

$$f(h'_{11},\ldots,h'_{NM}) \leqq f(h_{11},\ldots,h_{NM}),$$

d.h., H' ist optimal.

=> SE enthält mindestens einen optimalen Ablaufplan.

v. Analog zu iv.

vi. Dazu sei Beispiel 1.2 mit den Matrizen

$$P = \begin{pmatrix} 4 & 2 & 1 \\ 3 & 7 & 5 \\ 2 & 4 & 6 \end{pmatrix}$$

$$\text{und } Q = \begin{pmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \\ 1 & 2 & 3 \end{pmatrix}$$

betrachtet. Es soll bezüglich der Gesamtdurchlaufzeit der optimale Ablaufplan gesucht werden.

Wie in Kapitel 4 gezeigt wird, hat der optimale Ablaufplan folgendes Aussehen:

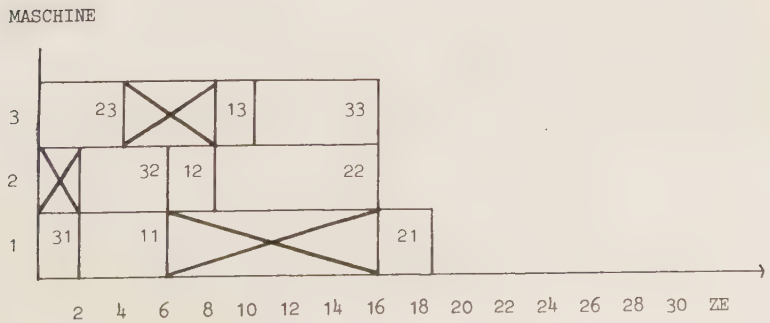



Abbildung 9

Der optimale Ablaufplan mit dem Zielfunktionswert 42 ist aktiv, also nicht unverzögert, da Operation 33 zum Zeitpunkt 6 zur Bearbeitung freigegeben ist und Maschine 3 zum Zeitpunkt 6 frei ist.

Der beste unverzögerte Ablaufplan bezüglich des Zielkriteriums „Minimierung der Durchlaufzeit" hat den Wert 43 und ist in Abbildung 10 dargestellt.

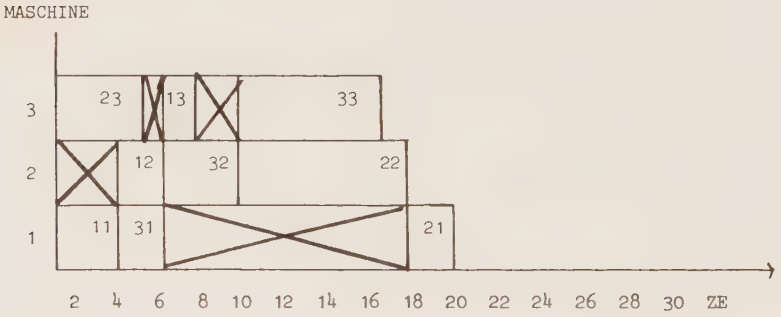



Abbildung 10

## 3.6. Untere Schranken für optimale Zielfunktionswerte

In den Abschnitten 3.2, 3.3 und 3.5 wurde die Erzeugung von semiaktiven, aktiven und unverzögerten Ablaufplänen diskutiert. Wurde ein Ablaufplan für ein Job-Shop-Problem ermittelt, so konnten für die drei erwähnten Zielsetzungen die zugehörigen Zielwerte ermittelt werden. Es können keine Aussagen darüber gemacht werden, wie weit die ermittelten Zielwerte von den optimalen Zielwerten entfernt sind.
In diesem Abschnitt wird versucht, untere Schranken für den optimalen Zielfunktionswert anzugeben. Jeder zulässige Ablaufplan bildet eine obere Schranke für den optimalen Zielwert.

GONZALEZ und SAHNI geben in [25] untere Schranken für den optimalen Wert der Zielfunktion bei der Minimierung der Durchlaufzeit und der Minimierung der Zykluszeit an.
Die erwähnten Autoren versuchen für den Quotienten aus dem schlechtest möglichen zulässigen Ablaufplan und dem optimalen Ablaufplan eine Schranke abzuleiten. Dazu betrachten sie sogenannte "busy"-Schedules, die dadurch gekennzeichnet sind, daß zu jedem Zeitpunkt vom Beginn der Planungsperiode bis zum Ende des Ablaufplanungsprozesses mindestens eine Maschine belegt ist. Insbesondere sind semiaktive und damit auch aktive und unverzögerte Ablaufpläne "busy"-Schedules.

**Satz 3.6:** (GONZALEZ/SAHNI [25]):
$GD^*$ sei der Wert einer optimalen Reihenfolge für ein Job-Shop-Problem mit M Maschinen und N Aufträgen, wobei die Gesamtdurchlaufzeit zu minimieren ist. GD sei der Wert irgendeiner "busy"-Schedule für dieses Problem. Dann gilt $GD/GD^* \leq N$ oder

$$(3\text{-}1) \quad LBGD^*1 := GD/N \leq GD^* \quad \text{und} \quad GD^* \leq GD.$$

**Beweis:** $L_I := \sum_{J=1}^{M} p_{IJ} \quad (I = 1,\ldots,N)$

$$T := \sum_{I=1}^{N} L_I$$

Für irgendeine beliebige "busy"-Schedule gilt dann für den Zielfunktionswert

$$GD \leqq N * T.$$

Für den optimalen Zielfunktionswert $GD^*$ gilt

$$GD^* \geqq T$$

$$\rightarrow GD/GD^* \leqq N$$

$GD^* \leqq GD$ gilt trivialerweise.

**Satz 3.7:** (GONZALEZ/SAHNI [25]):
$Z^*$ sei die Zykluszeit einer optimalen Reihenfolge für ein Job-Shop-Problem mit M Maschinen (M > 2) und N Aufträgen, wobei die Zykluszeit zu minimieren ist. Z sei der Wert irgendeiner "busy"-Schedule für dieses Problem. Dann gilt $Z/Z^* \leqq M$ oder

(3-2) $LBZ^*1 := Z/M \leqq Z^*$ und $Z^* \leqq Z$.

**Beweis:** $T := \sum_{I=1}^{N} L_I$

Für eine beliebige "busy"-Schedule ist

$$Z \leqq T.$$

Für eine optimale Schedule gilt:

$$Z^* \geqq T/M$$

$$\rightarrow Z/Z^* \leqq M$$

$Z^* \leqq Z$ gilt trivialerweise.

Satz 3.8:

$GB^*$ sei der Wert einer optimalen Reihenfolge für ein Job-Shop-Problem mit M Maschinen und N Aufträgen, wobei die Gesamtbelegungszeit zu minimieren ist. GB sei der Wert irgendeiner "busy"-Schedule für dieses Problem. Dann gilt $GB/GB^* \leqq M$ oder

(3-3) $\quad LBGB^*1 := GB/M \leqq GB^* \quad$ und $\quad GB^* \leqq GB.$

Beweis: Aus Satz 3.7 ist bekannt

$$Z \leqq T$$
$$\rightarrow GB \leqq M * Z \leqq M * T.$$

Weiterhin gilt

$$GB^* \geqq T$$
$$\rightarrow GB/GB^* \leqq M$$
$$\rightarrow GB/M \leqq GB^*.$$

$GB^* \leqq GB$ gilt trivialerweise.

Um die Güte der gewonnenen unteren Schranken zu demonstrieren, wird Beispiel 3.8 betrachtet, wobei als "busy"-Schedules unverzögerte Ablaufpläne konstruiert werden.

Beispiel 3.8:
In Beispiel 3.6 wurde für das Job-Shop-Problem aus Beispiel 1.2 ein unverzögerter Ablaufplan erzeugt, der folgende Zielwerte lieferte:

$$GD := 46,$$
$$Z := 23,$$
$$GB := 56.$$

Mit den Sätzen 3.6, 3.7 und 3.8 ergeben sich für die optimalen Zielwerte diese Abschätzungen:

$$15.33 \leqq GD^* \leqq 46,$$
$$7.66 \leqq Z^* \leqq 23,$$
$$18.66 \leqq GB^* \leqq 56.$$

Die angegebenen Intervalle sind recht groß für den optimalen Zielwert. Es wird im folgenden versucht, diese Intervalle möglichst zu verkleinern. Dabei muß darauf geachtet werden, daß die Bestimmung von unteren Schranken in polynomisch beschränkter Zeit geschieht.

Satz 3.9:
Eine untere Schranke für den optimalen Zielwert $Z^*$ eines Job-Shop-Problems mit N Aufträgen und M Maschinen ist (vgl. auch SEELBACH [54])

(3-4) $LBZ^*2 := \max \{LBAZ^*, LBMZ^*\}$, wobei

(3-5) $$LBAZ^* := \max \{ \sum_{J=1}^{M} p_{IJ} \mid I = 1,\ldots,N\} \text{ und}$$

(3-6) $$LBMZ^* := \max \{ \sum_{I=1}^{N} p_{IJ} \mid J = 1,\ldots,M\}.$$

Beweis: Sind noch keine Reihenfolgen auf den M Maschinen festgelegt, so bildet das Maximum der Zeilen- und Spaltensummen der Matrix P eine untere Schranke für den optimalen Zielwert $Z^*$.

LBAZ* ist dabei das Maximum der Zeilensummen, d.h. gleich dem Wert der Bearbeitungszeiten desjenigen Auftrags, der die größte Summe der Gesamtbearbeitungszeiten besitzt.

LBMZ* ist das Maximum der Spaltensummen, d.h. gleich dem maximalen Element unter den Werten, die jeweils eine untere Schranke für die Belegungszeit der M Maschinen bilden.

Die Bestimmung der Schranke (3-4) kann in polynomiell beschränkter Zeit geschehen, da die Bestimmung eines Maximums einer endlichen Menge durch die Mächtigkeit dieser Menge in der Zeitkomplexität beschränkt ist.

Eine weitere untere Schranke läßt sich über M Hilfsprobleme bestimmen. Dazu sei Maschine J betrachtet. Auf Maschine J müssen N Operationen ausgeführt werden, die mit einer gewissen Bearbeitungszeit $p_{IJ}$ verbunden sind. Desweiteren stehen die durchzuführenden Bearbeitungen unter Umständen erst zu verschiedenen Zeitpunkten zur Verfügung, da zuerst technologische Vorgängeroperationen abgeschlossen sein müssen. Eine Reihenfolge der N Bearbeitungen auf Maschine J ist nun so zu bestimmen, daß die Zykluszeit (dies ist im Ein-Maschinen-Fall äquivalent mit der Belegungszeit) minimal wird, wenn die N Operationen erst zu verschiedenen Zeitpunkten $a_{IJ}$ zur Verfügung stehen. Algorithmus 6 leistet dies.

Algorithmus 6:

*Name:* Verfahren zur Erzeugung einer optimalen Reihenfolge von N Bearbeitungen auf J Maschinen bezüglich Belegungszeit (Zykluszeit) bei im allgemeinen unterschiedlicher Ankunftszeit $a_{IJ}$.

*Voraussetzung:* J fix, $p_{IJ} \geq 0$, $a_{IJ} \geq 0$ (I = 1,...,N). $a_{IJ}$ ist die Summe der Bearbeitungszeiten der Vorgängeroperationen von IJ.

*Initialisierung:* Setze *DNM*(IJ) = 0 (I = 0,...,N).

*Iteration:*
1. Ordne in einer Menge S alle Operationen nach steigenden Ankunftsterminen an.
2. L = 1.
3. Wähle aus S den L-ten Index IJ aus.
4. *DNM*(IJ) = max {$a_{IJ}$,*DNM*([L-1],J)} + $p_{IJ}$
5. L = L + 1.

*STOPKRITERIUM:* Ist L > N? Wenn ja, dann STOP. Wenn nein, wiederhole 3. bis 5.

Satz 3.10:
Algorithmus 6 bestimmt eine optimale Reihenfolge in $O(N^2)$.

**Beweis:** Schritt 1 besteht aus einem aufsteigenden Sortierprozeß einer Menge S, die N Elemente enthält. Um das kleinste Element, d.h. diejenige Bearbeitung mit dem kleinsten $a_{IJ}$ zu finden, sind N-1 Vergleiche notwendig. Um das Element zu finden, das am zweitkleinsten ist, werden N-2 Vergleiche benötigt. Insgesamt werden (N-1)*(N-2)/2 Vergleiche benötigt. Die Schritte 3 bis 5 werden jeweils N-mal durchlaufen, wobei in Schritt 4 das Maximum zweier Größen zu bestimmen ist. Algorithmus 6 bestimmt eine optimale Reihenfolge in $O(N^2)$. Wird für den Sortierprozeß HEAPSORT verwendet (vgl. dazu MEHLHORN [46]), so läßt sich die Zeitkomplexität von Algorithmus 6 auf $O(N*\log_2 N)$ reduzieren.

Algorithmus 6 läßt sich auf alle M Maschinen anwenden. Dabei liefert Algorithmus 6 insgesamt M Werte für die gesuchten Durchlaufzeiten. Das Maximum dieser Werte bildet eine untere Schranke für die optimale Zykluszeit $Z^*$. Im folgenden sei diese untere Schranke mit $LBZ^*3$ bezeichnet.

Beispiel 3.9:

Für Beispiel 1.2 sollen LBZ*2 und LBZ*3 bestimmt werden.

$$P = \begin{pmatrix} 4 & 2 & 1 \\ 3 & 7 & 5 \\ 2 & 4 & 6 \end{pmatrix} \qquad Q = \begin{pmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \\ 1 & 2 & 3 \end{pmatrix}$$

LBZ*2 := max {max {7,15,12}, max {9,13,12}}
= max {15,13} = 15 ZE.

Bestimmung von LBZ*3:

J = 1

| IJ | $a_{IJ}$ | $p_{IJ}$ | DNM |
|---|---|---|---|
| 11 | 0 | 4 | 4 |
| 21 | 12 | 3 | 15 |
| 31 | 0 | 2 | 6 |

| IJ | $a_{IJ}$ | $p_{IJ}$ | DNM |
|---|---|---|---|
| 12 | 4 | 2 | 8 |
| 22 | 5 | 7 | 15 |
| 32 | 2 | 4 | 6 |

| IJ | $a_{IJ}$ | $p_{IJ}$ | DNM |
|---|---|---|---|
| 13 | 6 | 1 | 7 |
| 23 | 0 | 5 | 5 |
| 33 | 6 | 6 | 13 |

→ LBZ*3 = 15 ZE.

Allgemein läßt sich für die drei erwähnten unteren Schranken folgender Satz herleiten:

Satz 3.11:

i. LBZ*2 $\geq$ LBZ*1, falls LBZ*2 = LBMZ*.
ii. LBZ*3 $\geq$ LBZ*2, falls LBZ*2 = LBMZ*.

Beweis:

i. Nach Voraussetzung gilt LBZ*2 = LBMZ*.

→ $M * LBZ^*2 \geq T := \sum_{I=1}^{N} \sum_{J=1}^{M} p_{IJ}$,

da aber $Z \leq T$,

→ $M * LBZ^*2 \geq Z$,

→ $LBZ^*2 \geq \frac{Z}{M} = LBZ^*1$

ii. Der jeweils größte Endtermin, der mit Algorithmus 6 für alle M Maschinen erzeugt wird, sei mit $DNM(KJ)$ $(J = 1,\ldots,M)$ bezeichnet.

$$\rightarrow LBZ^*3 = \max_{J=1,\ldots,M} \{DNM(KJ)\}$$

$$LBZ^*3 \geq LBAZ^*, \text{ da } \max_{J=1,\ldots,M} \{DNM(KJ)\}$$

$$\geq \max \{ \sum_{J=1}^{M} p_{IJ} \mid I = 1,\ldots,N\}.$$

Entsprechend gilt $LBZ^*3 \geq LBMZ^*$. Zusammen gilt also $LBZ^*3 \geq LBZ^*2$.

Satz 3.12:

Eine untere Schranke für den optimalen Zielwert $GB^*$ (Belegungszeit) eines Job-Shop-Problems mit N Aufträgen und M Maschinen ist

$$(3\text{-}7) \qquad LBGB^*2 := \sum_{J=1}^{M} \sum_{I=1}^{N} p_{IJ}.$$

Beweis: Die Gesamtbelegungszeit aller M Maschinen ist definiert als die Summe der Leerzeiten und die Summe der Bearbeitungszeiten. Wird die Summe der Leerzeiten vernachlässigt, so ergibt sich $LBGB^*2$ als untere Schranke für den optimalen Zielwert $GB^*$.

Eine weitere untere Schranke läßt sich mit Hilfe von Algorithmus 6 bestimmen.

Satz 3.13:

Die Summe aller M Endtermine, die mit Hilfe von Algorithmus 6 gefunden werden kann, bildet eine untere Schranke $LBGB^*3$ für den optimalen Zielwert $GB^*$.

**Beweis:** Jeder gefundene Endtermin eines Ein-Maschinen-Problems bildet eine untere Schranke für die Belegungszeit dieser Maschine. Die Summe aller M Endtermine bildet dann also eine untere Schranke für den optimalen Zielwert $GB^*$.

**Beispiel 3.10:**
Für Beispiel 1.2 soll $LBGB^*2$ und $LBGB^*3$ bestimmt werden (vgl. für P und Q auch Beispiel 3.9).

$$LBGB^*2 := 4 + 2 + 1 + 3 + 7 + 5 + 2 + 4 + 6 = 34,$$
$$LBGB^*3 := 15 + 15 + 13 = 43.$$

Allgemein läßt sich über die drei erwähnten unteren Schranken der nachstehende Satz formulieren.

**Satz 3.14:**

i. $LBGB^*2 \geq LBGB^*1.$

ii. $LBGB^*3 \geq LBGB^*2.$

**Beweis:**

i. Aus Satz 3.8 ist bekannt

$$GB \leq M * T$$
$$\rightarrow \underbrace{GB/M}_{LBGB^*1} \leq T := \underbrace{\sum_{I=1}^{N} \sum_{J=1}^{M} p_{IJ}}_{LBGB^*2}$$

ii. $$LBGB^*3 := \sum_{J=1}^{M} DNM(KJ) \geq \sum_{J=1}^{M} \sum_{I=1}^{N} p_{IJ}$$
$$:= LBGB^*2$$

Für die Durchlaufzeit kann die folgende Schranke hergeleitet werden.

Satz 3.15:
Eine untere Schranke für den optimalen Zielwert GD* (Durchlaufzeit) eines Job-Shop-Problems mit N Aufträgen und M Maschinen ist

$$(3\text{-}8) \quad LBGD^*2 := \sum_{J=1}^{M} \sum_{I=1}^{N} p_{IJ}.$$

**Beweis:** Analog zu Satz 3.12.

Algorithmus 6 legt die Vermutung nahe, daß mit seiner Hilfe auch für die Durchlaufzeit untere Schranken gefunden werden können. Dies ist im allgemeinen nicht der Fall, wie Beispiel 3.11 zeigt.

Beispiel 3.11:
Gegeben sei das Job-Shop-Problem mit den Matrizen

$$P = \begin{pmatrix} 10 & 4 \\ 5 & 1 \end{pmatrix} \qquad Q = \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix}$$

Wendet man Algorithmus 6 auf Maschine 1 an, so ergibt sich:

| IJ | $a_{IJ}$ | $p_{IJ}$ | DNM |
|---|---|---|---|
| 11 | 0 | 10 | 10 |
| 21 | 1 | 5 | 15 |

Als Durchlaufzeit für die Operationen 11 und 21 ergibt sich: GD = (0+10) + (9+5) = 24 ZE.

Wird Operation 21 vor Operation 11 eingeplant und damit gegen die Vorschriften von Algorithmus 6, ergibt sich eine Durchlaufzeit D = (6+10) + (0+5) = 21 ZE. Algorithmus 6 kann also nicht zur Konstruktion unterer Schranken für die optimale Durchlaufzeit benutzt werden.

Es entsteht das Problem, N Aufträge mit unterschiedlichen Ankunftszeiten auf einer Maschine so einzuplanen, daß die Durchlaufzeit minimal wird. Dafür existiert jedoch kein polynomiell beschränkter Algorithmus.

RINNOOY KAN zeigt in [51, S. 81, Theorem 4.25], daß das oben erwähnte Problem $\mathcal{NP}$-vollständig ist.

Es läßt sich jedoch ein polynomiell beschränkter Algorithmus angeben, der eine untere Schranke für die Durchlaufzeit im Ein-Maschinen-Fall angibt. Dabei wird zugelassen, daß die Maschine beliebig viele Aufträge zur gleichen Zeit bearbeiten kann. Jede Operation wird eingeplant, wenn sie verfügbar ist. Dadurch entstehen unter Umständen Mehrfachbearbeitungen zu einem Zeitpunkt. Angenommen, es werden n Operationen ($n > 1$, $n \in \mathbb{N}$) im Intervall

$[t,t+1]$, $0 \leq t \leq T := \sum_{I=1}^{N} \sum_{J=1}^{M} p_{IJ}$ eingeplant, so können dadurch n-1 ZE Wartezeiten entstehen, wenn eine Maschine gleichzeitig nur einen Auftrag bearbeiten kann.

**Beispiel 3.12:**

Es soll eine untere Schranke für die Durchlaufzeit der Operationen 12, 22 und 32 in Beispiel 1.2 bestimmt werden.

| IJ | $a_{IJ}$ | $p_{IJ}$ |
|---|---|---|
| 12 | 4 | 2 |
| 22 | 5 | 7 |
| 32 | 2 | 4 |

Es existieren insgesamt sechs Möglichkeiten, die Reihenfolge der Operationen 12, 22 und 32 einzuplanen. Die minimale Durchlaufzeit liefert die Reihenfolge 32, 12 und 22

mit einer Wartezeit von 5 ZE und einer Bearbeitungszeit von 13 ZE.

Wird mit dem Verfahren, wie es oben angedeutet ist, eine untere Schranke für die optimale Durchlaufzeit gesucht, so ergibt sich folgende Situation, die in Abbildung 11 dargestellt ist.

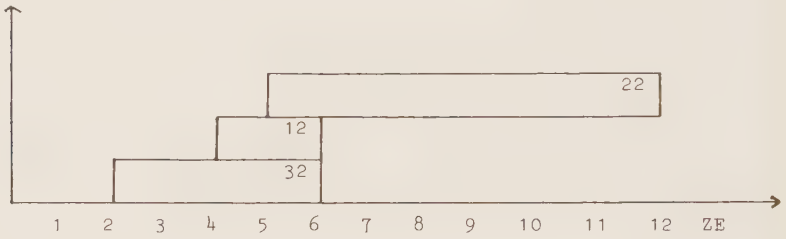



Abbildung 11

Im Zeitintervall [4,5] werden zwei Aufträge bearbeitet. Folglich entsteht mindestens eine Wartezeit von 1 ZE. Im Intervall [5,6] arbeiten drei Aufträge. Es entsteht eine Wartezeit von mindestens 2 ZE. Daraus ergibt sich eine untere Schranke für die Durchlaufzeit von 3 ZE + 13 ZE = 16 ZE.

Das oben angedeutete Verfahren soll formal in Algorithmus 7 dargestellt werden und die Zeitkomplexität des Verfahrens untersucht werden.

Algorithmus 7:

*Name:* Verfahren zur Bestimmung einer unteren Schranke für die Wartezeit von N Operationen auf einer Maschine.

*Voraussetzung:* J fix, $p_{IJ} \geq 0$ und ganzzahlig, $a_{IJ} \geq 0$ (I = 1,...,N), wobei $a_{IJ}$ die Summe der Bearbeitungszeiten ist, die technologisch vor der Bearbeitung auf Maschine J anfallen.

*Initialisierung:* Setze *DNM*(IJ) = 0 (I = 1,...,N).

*Iteration:*

1. L = 1.
2. *DNM*(LJ) = $a_{LJ} + p_{LJ}$.
3. L = L + 1. Wenn L > N, dann gehe zu 4., anderenfalls zu 2.
4. TMIN = $\min_{I=1,\ldots,N} \{a_{IJ}\}$

   TMAX = $\max_{I=1,\ldots,N} \{a_{IJ}+p_{IJ}\}$.
5. ZAEHLER = TMIN; ANZAHL = 0.
6. FOR I. = 1(1)N DO.

   Wenn ZAEHLER $\in [a_{IJ}, a_{IJ}+p_{IJ}]$, dann ANZAHL = ANZAHL +1.
7. Falls Anzahl = 0, gehe zu 9.
8. WARTEZEIT = ANZAHL -1.
9. ZAEHLER = ZAEHLER +1; ANZAHL = 0.

*STOPKRITERIUM:*

10. Falls ZAEHLER > TMAX, dann STOP, anderenfalls gehe zu 6.

**Satz 3.16:**

Algorithmus 7 bestimmt in O((TMAX-TMIN)*N) eine untere Schranke für die Durchlaufzeit von N Operationen auf einer Maschine J.

**Beweis:** Schritte 1, 2 und 3 benötigen O(N) Schritte. Schritt 4 ist ebenfalls durch O(N) beschränkt. Schritt 6 benötigt O(N) Schritte. Insgesamt wird Schritt 6 (TMAX-TMIN)-mal durchlaufen, d.h., die Zeitkomplexität beträgt O((TMAX-TMIN)*N).

**Bemerkung 3.6:**

Algorithmus 7 arbeitet nur richtig, wenn die Bearbeitungsdauern $p_{IJ}$ ganzzahlig sind. Der Algorithmus arbeitet in einer „pseudopolynomiellen" Zeitkomplexität, da die Laufzeit

von Algorithmus 7 nicht nur von M oder N abhängt, sondern auch noch von der Differenz TMAX - TMIN. Der Begriff der „pseudopolynomiellen" Zeitkomplexität wurde von LAWLER [39] eingeführt. Eine untere Schranke für die Durchlaufzeit ist die Summe der Bearbeitungszeiten auf den M Maschinen und dem Maximum der Wartezeiten, die auf den M Maschinen entstehen können. Der zweite Teil dieser Aussage läßt sich sofort daraus folgern, daß Algorithmus 7 für jede der M Maschinen eine untere Schranke für die Wartezeit bestimmt. Insbesondere gilt dann für das Maximum der Wartezeiten, daß es eine untere Schranke für die Gesamtwartezeit darstellt.

Der größte Nachteil in Algorithmus 7 besteht in der „pseudopolynomiellen" Laufzeit. Algorithmus 7a beschreibt die Konstruktion eines polynomiell beschränkten Algorithmus zur Bestimmung einer unteren Schranke für die Durchlaufzeit. Diese Laufzeit kann allerdings im allgemeinen nur durch einen Güteverlust der unteren Schranke erzwungen werden.

Algorithmus 7a:

*Name:* Verfahren zur Bestimmung einer unteren Schranke für die Wartezeit von N Aufträgen auf einer Maschine.

*Voraussetzung:* J fix, $p_{IJ} \geq 0$, $a_{IJ} \geq 0$ (I = 1,...,N).

*Initialisierung:* Setze *DNM*(IJ) = 0 (I = 1,...,N).

*Iteration:*

1. L = 1.
2. *DNM*(LJ) = $a_{LJ} + p_{LJ}$.
3. L = L + 1. Wenn L > N, dann gehe zu 4., anderenfalls zu 2.
4. Ermittle mit Algorithmus 6 die maximale Belegungszeit der Maschine J.
5. Bilde die Differenz zwischen der maximalen Belegungszeit, die mit Algorithmus 6 ermittelt wird, und dem maximalen Endtermin, der mit den Schritten 1, 2 und 3 ermittelt wurde.

Satz 3.17:
Die Laufzeit von Algorithmus 7a beträgt $O(N^2)$.

Beweis: Die Schritte 1, 2 und 3 besitzen die Zeitkomplexität $O(N)$. Schritt 4 benötigt nach Satz 3.10 $O(N^2)$.

Bemerkung 3.7:
In Algorithmus 7a wird die Differenz zwischen frühest möglichem Endtermin aller N Aufträge und dem Maximum der Termine gebildet, wenn nur technologische Reihenfolgen berücksichtigt werden. Da diese Aussage für alle M Maschinen gilt, bildet das Maximum dieser M Differenzen zusammen mit den Gesamtbearbeitungszeiten eine untere Schranke für die Gesamtdurchlaufzeit. Die durch Algorithmus 7 gefundene untere Schranke für die Wartezeit auf Maschine J ($J \in \{1,\ldots,M\}$) ist im allgemeinen schwächer als die durch Algorithmus 7 bestimmte untere Schranke, wie im Beispiel 3.13 gezeigt wird.

Beispiel 3.13:
Für Beispiel 1.2 soll eine untere Schranke mit Hilfe von Algorithmus 7a für die Gesamtdurchlaufzeit angegeben werden.

Die Summe der Bearbeitungszeiten beträgt 34 ZE.

| IJ | $a_{IJ}$ | $p_{IJ}$ | DNM(IJ) |
|---|---|---|---|
| 11 | 0 | 4 | 4 |
| 21 | 12 | 3 | 15 |
| 31 | 0 | 2 | 2 |

| IJ | $a_{IJ}$ | $p_{IJ}$ | DNM(IJ) |
|---|---|---|---|
| 12 | 4 | 2 | 6 |
| 22 | 5 | 7 | 12 |
| 32 | 2 | 4 | 6 |

| IJ | $a_{IJ}$ | $p_{IJ}$ | DNM(IJ) |
|---|---|---|---|
| 13 | 6 | 1 | 7 |
| 23 | 0 | 5 | 5 |
| 33 | 6 | 6 | 12 |

Auf Maschine 1 entstehen 2 ZE Wartezeit.
Auf Maschine 2 entstehen 3 ZE Wartezeit.
Auf Maschine 3 entsteht 1 ZE Wartezeit.

Eine untere Schranke für die Gesamtdurchlaufzeit $GD^*$ ist $\max\{2,3,1\} + 34 = 37$ ZE.

Mit Algorithmus 7a ergeben sich die drei Differenzen:
Maschine 1: 15 ZE - 15 ZE = 0 ZE,
Maschine 2: 15 ZE - 12 ZE = 3 ZE,
Maschine 3: 13 ZE - 12 ZE = 1 ZE.
Algorithmus 7 ermittelt für Maschine 1 eine Wartezeit von 2 ZE, während Algorithmus 7a eine Wartezeit von 0 ZE anzeigt. Allerdings liefern beide Algorithmen für Beispiel 1.2 die gleiche untere Schranke für die Gesamtdurchlaufzeit von 37 ZE.

## 4. Gemischt-ganzzahlige Ablaufplanungsmodelle und ihre Lösungsmöglichkeiten

### 4.1. Das MANNE-Modell

1960 entwickelte MANNE in [44] ein gemischt-ganzzahliges Optimierungsmodell zur Ablaufplanung. Dieser Ansatz dürfte bis heute das beste analytische Modell zur Bestimmung optimaler Ablaufpläne sein. Für eine weitere Diskussion gemischt-ganzzahliger, rein ganzzahliger Ablaufplanungsmodelle sei auf SEELBACH [54] verwiesen. Eine nichtlineare Modellformulierung wurde u.a. von HUCKERT [31] vorgeschlagen, der mit dem Algorithmus von JAHN [32] das zugehörige nichtlineare Modell löst.

Das Ablaufplanungsproblem ist - unabhängig von der Zielfunktion - mindestens zwei Arten von Restriktionen unterworfen:

i. Die technologischen Maschinenfolgebedingungen
Die technologischen Reihenfolgebedingungen, die durch die Matrix Q beschrieben werden, müssen eingehalten werden. Daher gilt:

$$(4\text{-}1) \qquad h_{I[J]} + p_{I[J]} \leq h_{I[J+1]} \qquad \begin{matrix} (I = 1,\ldots,N; \\ J = 1,\ldots,M-1). \end{matrix}$$

ii. Die Auftragsfolgebedingungen
Durch die Auftragsfolgebedingungen wird verhindert, daß eine Maschine gleichzeitig durch mehr als einen Auftrag in Anspruch genommen wird. Um sicherzustellen, daß eine Bearbeitung des Auftrags I auf der Maschine J frühestens nach Abschluß der Bearbeitung eines Auftrags K auf derselben Maschine erfolgen kann, muß gelten:

$$(4\text{-}2) \qquad h_{KJ} + p_{KJ} \leq h_{IJ} \qquad I \neq K \qquad \begin{matrix} I,K \in \{1,\ldots,N\}, \\ J \in \{1,\ldots,M\}. \end{matrix}$$

Nun kann aber Auftrag I auf Maschine J auch vor Auftrag K eingeplant werden. In diesem Falle gilt dann:

$$(4\text{-}3) \quad h_{IJ} + p_{IJ} \leqq h_{KJ} \quad I \neq K \qquad \begin{matrix} I,K \in \{1,\ldots,N\}, \\ J \in \{1,\ldots,M\}. \end{matrix}$$

Von den beiden Auftragsfolgebedingungen (4-2) und (4-3) kann aber nur *eine* Bedingung gelten. MANNE erreicht dies in [44], indem er Binärvariablen und eine obere Schranke UB einführt. Die Auftragsfolgebedingungen für die Maschine J bezüglich der Aufträge I und K lauten dann:

$$(4\text{-}4) \quad h_{KJ} + p_{KJ} \leqq h_{IJ} + UB*(1-\delta_{I,K,J})$$

$$(4\text{-}5) \quad h_{IJ} + p_{IJ} \leqq h_{KJ} + UB*\delta_{I,K,J}$$

$$\text{mit } \delta_{I,K,J} \in \{0,1\},\ I \neq K \qquad \begin{matrix} I,K \in \{1,\ldots,N\}, \\ J \in \{1,\ldots,M\}. \end{matrix}$$

Bemerkung 4.1:

Die obigen beiden Ungleichungen entsprechen in dem disjunktiven Graphen G den disjunktiven Pfeilen der Menge D. Für alle Aufträge lauten die Auftragsfolgebedingungen:

$$(4\text{-}6) \quad h_{IJ} + p_{IJ} \leqq h_{I+L,J} + UB*(1-\delta_{I,L,J})$$

$$(4\text{-}7) \quad h_{I+L,J} + p_{I+L,J} \leqq h_{IJ} + UB*\delta_{I,L,J}$$

$$(J = 1,\ldots,M;\ I = 1,\ldots,N-1;\ L = 1,\ldots,N-I).$$

Bemerkung 4.2:

Die Gesamtheit aller obigen Restriktionen beschreibt die Menge D der disjunktiven Pfeile. Wird den Binärvariablen genau ein Wert 0 oder 1 zugewiesen, so wird eine der Restriktionen (4-6) oder (4-7) redundant.

### 4.1.1. Das MANNE-Modell für die Zielsetzung „Minimierung der Gesamtdurchlaufzeit"

$$(4\text{-}8) \quad \min \quad \sum_{I=1}^{N} (h_{I[M]} + p_{I[M]})$$

u.d.N.

$$h_{I[J]} + p_{I[J]} \leq h_{I[J+1]} \quad (I = 1,\ldots,N;\ J = 1,\ldots,M-1)$$

$$h_{IJ} + p_{IJ} \leq h_{I+L,J} + UB*(1-\delta_{I,L,J})$$

$$h_{I+L,J} + p_{I+L,J} \leq h_{IJ} + UB*\delta_{I,L,J}$$

$$(J = 1,\ldots,M;\ I = 1,\ldots,N-1;\ L = 1,\ldots,N-I)$$

$$h_{IJ} \geq 0 \quad (I = 1,\ldots,N;\ J = 1,\ldots,M);$$

$$\delta_{I,L,J} \in \{0,1\} \quad (J = 1,\ldots,M;\ I = 1,\ldots,N-1;\ L = 1,\ldots,N-I)$$

Die Zahl der Variablen und der Restriktionen dieses gemischt-ganzzahligen Entscheidungsmodells ist der nachfolgenden Tabelle zu entnehmen:

| Zielsetzung | Variablenzahl | Auftragsfolgebedingungen | Maschinenfolgebedingungen |
|---|---|---|---|
| Gesamtdurchlaufzeitminimierung | N*M + N*(N-1)*M/2 | N*(N-1)*M | N*(M-1) |

Dieses Entscheidungsmodell besitzt N*M stetige Entscheidungsvariablen und N*(N-1)*M/2 Binärvariablen.

**Beispiel 4.1:**

Für Beispiel 1.2 mit den Matrizen

$$P = \begin{pmatrix} 4 & 2 & 1 \\ 3 & 7 & 5 \\ 2 & 4 & 6 \end{pmatrix} \text{ und } Q = \begin{pmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \\ 1 & 2 & 3 \end{pmatrix}$$

soll zunächst das MANNE-Modell dargestellt werden und dann graphentheoretisch interpretiert werden.

$$
\begin{aligned}
(4\text{-}9)\quad \min\quad & h_{13} + h_{21} + h_{33} + 10 \\
\text{u.d.N.}\quad & h_{11} + 4 \leqq h_{12} \\
& h_{12} + 2 \leqq h_{13} \\
& h_{23} + 5 \leqq h_{22} \\
& h_{22} + 7 \leqq h_{21} \\
& h_{31} + 2 \leqq h_{32} \\
& h_{32} + 4 \leqq h_{33} \\
& h_{11} + 4 \leqq h_{21} + UB*(1-\delta_{121}) \\
& h_{21} + 3 \leqq h_{11} + UB*\delta_{121} \\
& h_{11} + 4 \leqq h_{31} + UB*(1-\delta_{131}) \\
& h_{31} + 2 \leqq h_{11} + UB*\delta_{131} \\
& h_{12} + 2 \leqq h_{22} + UB*(1-\delta_{122}) \\
& h_{22} + 7 \leqq h_{12} + UB*\delta_{122} \\
& h_{12} + 2 \leqq h_{32} + UB*(1-\delta_{132}) \\
& h_{32} + 4 \leqq h_{12} + UB*\delta_{132} \\
& h_{22} + 7 \leqq h_{32} + UB*(1-\delta_{232}) \\
& h_{32} + 4 \leqq h_{22} + UB*\delta_{232} \\
& h_{13} + 1 \leqq h_{23} + UB*(1-\delta_{123}) \\
& h_{23} + 5 \leqq h_{13} + UB*\delta_{123} \\
& h_{13} + 1 \leqq h_{33} + UB*(1-\delta_{133}) \\
& h_{33} + 6 \leqq h_{13} + UB*\delta_{133} \\
& h_{23} + 5 \leqq h_{33} + UB*(1-\delta_{233}) \\
& h_{33} + 6 \leqq h_{23} + UB*\delta_{233} \\
& \\
& h_{11},\ldots,h_{33} \geqq 0
\end{aligned}
$$

Modell (4-9) wird in Abbildung 12 graphisch dargestellt.

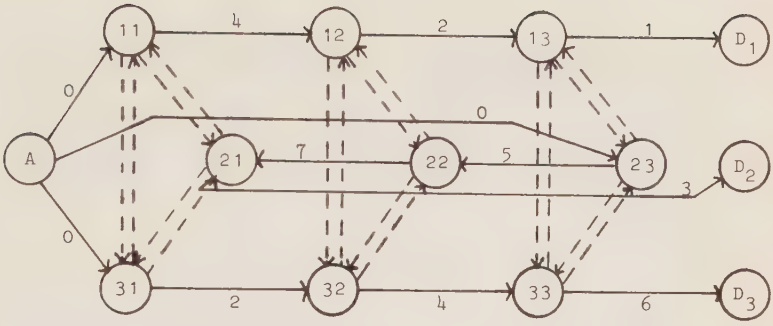



Abbildung 12

Bemerkung 4.3:

Knoten A in Abbildung 12 stellt den Beginn der Planungsperiode dar. Die drei Pfeile mit der Bewertung 0, die von A ausgehen, zeigen an, daß die erste Bearbeitung jedes Auftrags sofort begonnen werden kann.

Die Knoten $D_1$, $D_2$ und $D_3$ stellen die Endpunkte der Bearbeitungen der drei Aufträge dar.

Die Minimierungsvorschrift in (4-9) läßt im disjunktiven Graphen G in Abbildung 12 folgende Interpretation zu: Suche im Graphen G durch Entfernen jeweils eines Pfeiles der disjunktiven Pfeilpaare diejenigen Wege von A zu $D_1$, $D_2$ und $D_3$ mit der Eigenschaft, daß

i. die technologischen Reihenfolgen und Auftragsfolgen eingehalten werden,
ii. kein Zyklus im Graphen F ist und
iii. die Längen minimal werden.

Wege, die i. und ii. erfüllen, heißen längste Wege von A zu den entsprechenden Endknoten. Minimale Wege zu diesen Knoten heißen auch minimaximale Wege.

### 4.1.2. Das MANNE-Modell für die Zielsetzung „Minimierung der Zykluszeit"

In Abschnitt 1.2 wurde die Zykluszeit als das Maximum der Endtermine der letzten Bearbeitungen aller Aufträge definiert.

$$(4\text{-}10) \quad Z := \max \{h_{1[M]} + p_{1[M]}, \ldots, h_{N[M]} + p_{N[M]}\}$$

In dieser Form läßt sich die Minimierung der Zykluszeit nicht in das MANNE-Modell implementieren. Da jeder der N Endtermine das maximale Element in (4-10) bilden kann, muß zur Formulierung von Restriktionen, die linear sein sollen, jeder Endtermin durch Z nach oben beschränkt werden, d.h.

$$(4\text{-}11) \quad h_{I[M]} + p_{I[M]} \leqq Z \qquad (I = 1,\ldots,N).$$

Also lautet das MANNE-Modell:

$$(4\text{-}12) \quad \min \quad Z$$

$$\text{u.d.N.} \quad h_{I[J]} + p_{I[J]} \leqq h_{I[J+1]} \qquad (I = 1,\ldots,N;\ J = 1,\ldots,M-1)$$

$$h_{IJ} + p_{IJ} \leqq h_{I+L,J} + UB*(1-\delta_{I,L,J})$$

$$h_{I+L,J} + p_{I+L,J} \leqq h_{IJ} + UB*\delta_{I,L,J}$$

$$(J = 1,\ldots,M;\ I = 1,\ldots,N-1;\ L = 1,\ldots,N-I)$$

$$h_{I[M]} + p_{I[M]} \leqq Z \qquad (I = 1,\ldots,N)$$

$$Z,\ h_{IJ} \geqq 0 \qquad (I = 1,\ldots,N;\ J = 1,\ldots,M)$$

$$\delta_{I,L,J} \in \{0,1\} \qquad (J = 1,\ldots,M;\ I = 1,\ldots,N-1;\ L = 1,\ldots,N-I)$$

Die folgende Tabelle gibt die Zahl der Restriktionen und Variablen für dieses gemischt ganzzahlige Entscheidungsproblem an.

| Zielsetzung | Variablenzahl | Auftragsfolgebedingungen | Maschinenfolgebedingungen | Zyklusbeschränkungen |
|---|---|---|---|---|
| Zykluszeitminimierung | N*M + 1 + N*(N-1)*M/2 | N*(N-1)*M | N*(M-1) | N |

**Beispiel 4.2:**

Für Beispiel 1.2 wird das MANNE-Modell mit der Zielsetzung „Minimierung der Zykluszeit" gebildet.

(4-13) min Z

u.d.N.

$$h_{11} + 4 \leqq h_{12}$$
$$h_{12} + 2 \leqq h_{13}$$
$$h_{23} + 5 \leqq h_{22}$$
$$h_{22} + 7 \leqq h_{21}$$
$$h_{31} + 2 \leqq h_{32}$$
$$h_{32} + 4 \leqq h_{33}$$
$$h_{11} + 4 \leqq h_{21} + UB*(1-\delta_{121})$$
$$h_{21} + 3 \leqq h_{11} + UB*\delta_{121}$$
$$h_{11} + 4 \leqq h_{31} + UB*(1-\delta_{131})$$
$$h_{31} + 2 \leqq h_{11} + UB*\delta_{131}$$
$$h_{12} + 2 \leqq h_{22} + UB*(1-\delta_{231})$$
$$h_{22} + 7 \leqq h_{12} + UB*\delta_{231}$$
$$h_{12} + 2 \leqq h_{32} + UB*(1-\delta_{132})$$
$$h_{32} + 4 \leqq h_{12} + UB*\delta_{132}$$
$$h_{22} + 7 \leqq h_{32} + UB*(1-\delta_{232})$$
$$h_{32} + 4 \leqq h_{22} + UB*\delta_{232}$$
$$h_{13} + 1 \leqq h_{23} + UB*(1-\delta_{123})$$
$$h_{23} + 5 \leqq h_{13} + UB*\delta_{123}$$
$$h_{13} + 1 \leqq h_{33} + UB*(1-\delta_{133})$$
$$h_{33} + 6 \leqq h_{13} + UB*\delta_{133}$$
$$h_{23} + 5 \leqq h_{33} + UB*(1-\delta_{233})$$
$$h_{33} + 6 \leqq h_{23} + UB*\delta_{233}$$

$$h_{13} + 1 \leqq Z$$
$$h_{21} + 3 \leqq Z$$
$$h_{33} + 6 \leqq Z$$

$$h_{11},\ldots,h_{33},Z \geqq 0$$

Modell (4-13) wird in Abbildung 13 verdeutlicht.

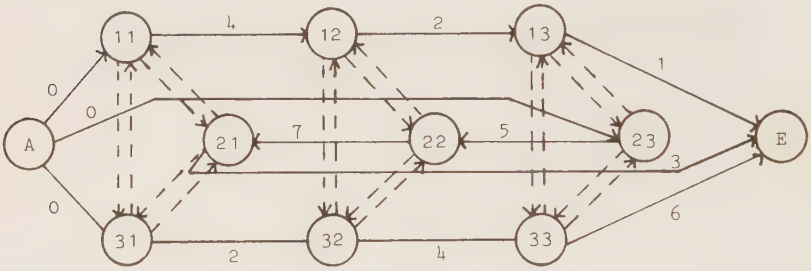



Abbildung 13

**Bemerkung 4.4:**
Abbildung 13 stellt ein disjunktives Netzwerk dar, das eine Quelle A und eine Senke E besitzt.

### 4.1.3. Das MANNE-Modell für die Zielsetzung „Minimierung der Belegungszeit"

In Kapitel 1.2 wurde die Belegungszeit $B_J$ einer Maschine J als das Maximum der Endtermine $h_{IJ} + p_{IJ}$ $(I = 1,\ldots,N)$ interpretiert. Da jeder der N Endtermine das maximale Element in der Menge der Endtermine ist, müssen für jede Maschine J N Beschränkungen der Form

$$(4\text{-}14) \quad h_{IJ} + p_{IJ} \leqq B_J \quad (J = 1,\ldots,M;\ I = 1,\ldots,N)$$

eingeführt werden.

Also lautet das MANNE-Modell für die Zielsetzung „Minimierung der Belegungszeit"

$$(4\text{-}15) \quad \min \quad \sum_{J=1}^{M} B_J$$

u.d.N.

$$h_{I[J]} + p_{I[J]} \leqq h_{I[J+1]} \quad \begin{matrix}(I = 1,\ldots,N;\\ J = 1,\ldots,M-1)\end{matrix}$$

$$h_{IJ} + p_{IJ} \leqq h_{I+L,J} + UB*(1-\delta_{I,L,J})$$

$$h_{I+L,J} + p_{I+L,J} \leqq h_{IJ} + UB*\delta_{I,L,J}$$

$$(J = 1,\ldots,M;\ I = 1,\ldots,N-1;\ L = 1,\ldots,N-I)$$

$$h_{IJ} + p_{IJ} \leqq B_J \qquad \begin{matrix}(J = 1,\ldots,M;\\ I = 1,\ldots,N)\end{matrix}$$

$$h_{IJ} \geqq 0 \qquad \begin{matrix}(I = 1,\ldots,N;\\ J = 1,\ldots,M)\end{matrix}$$

$$B_J \geqq 0 \qquad (J = 1,\ldots,M)$$

$$\delta_{I,L,J} \in \{0,1\} \quad \begin{matrix}(J = 1,\ldots,M;\ I = 1,\ldots,N-1;\\ L = 1,\ldots,N-I)\end{matrix}$$

Die Zahl der Variablen und Restriktionen dieses gemischt-ganzzahligen Entscheidungsmodells ist der folgenden Tabelle zu entnehmen:

| Zielsetzung | Variablenzahl | Auftragsfolgebedingungen | Maschinenfolgebedingungen | Belegungszeitbedingungen |
|---|---|---|---|---|
| Gesamtbelegungszeit | N*M + M + N*(N-1)*M/2 | N*(N-1)*M | N*(M-1) | M*N |

**Beispiel 4.3:**

Für Beispiel 1.2 wird das MANNE-Modell mit der Zielsetzung „Minimierung der Belegungszeit" gebildet.

$$
\begin{aligned}
(4\text{-}16)\quad \min\quad & B_1 + B_2 + B_3 \\
\text{u.d.N.}\quad & h_{11} + 4 \leqq h_{12} \\
& h_{12} + 2 \leqq h_{13} \\
& h_{23} + 5 \leqq h_{22} \\
& h_{22} + 7 \leqq h_{21} \\
& h_{31} + 2 \leqq h_{32} \\
& h_{32} + 4 \leqq h_{33} \\
& h_{11} + 4 \leqq h_{21} + UB*(1-\delta_{121}) \\
& h_{21} + 3 \leqq h_{11} + UB*\delta_{121} \\
& h_{11} + 4 \leqq h_{31} + UB*(1-\delta_{131}) \\
& h_{31} + 2 \leqq h_{11} + UB*\delta_{131} \\
& h_{12} + 2 \leqq h_{22} + UB*(1-\delta_{231}) \\
& h_{22} + 7 \leqq h_{12} + UB*\delta_{231} \\
& h_{12} + 2 \leqq h_{32} + UB*(1-\delta_{132}) \\
& h_{32} + 4 \leqq h_{12} + UB*\delta_{132} \\
& h_{22} + 7 \leqq h_{32} + UB*(1-\delta_{232}) \\
& h_{32} + 4 \leqq h_{22} + UB*\delta_{232} \\
& h_{13} + 1 \leqq h_{23} + UB*(1-\delta_{123}) \\
& h_{23} + 5 \leqq h_{13} + UB*\delta_{123} \\
& h_{13} + 1 \leqq h_{33} + UB*(1-\delta_{133}) \\
& h_{33} + 6 \leqq h_{13} + UB*\delta_{133} \\
& h_{23} + 5 \leqq h_{33} + UB*(1-\delta_{233}) \\
& h_{33} + 6 \leqq h_{23} + UB*\delta_{233} \\
& h_{11} + 4 \leqq B_1 \\
& h_{21} + 3 \leqq B_1 \\
& h_{31} + 2 \leqq B_1 \\
& h_{12} + 2 \leqq B_2 \\
& h_{22} + 7 \leqq B_2 \\
& h_{32} + 4 \leqq B_2 \\
& h_{13} + 1 \leqq B_3 \\
& h_{23} + 5 \leqq B_3 \\
& h_{33} + 6 \leqq B_3 \\
& \\
& h_{11},\ldots,h_{33},B_1,B_2,B_3 \geqq 0
\end{aligned}
$$

Modell (4-16) wird in Abbildung 14 verdeutlicht.

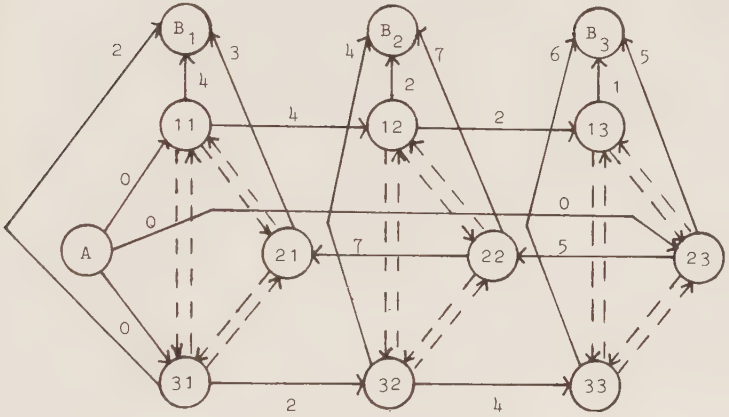



Abbildung 14

Bemerkung 4.5:

Die Knoten $B_1$, $B_2$ und $B_3$ stellen im disjunktiven Graphen die Endpunkte der drei Maschinen dar, in denen sie aufhören zu arbeiten. Da jeder der drei Aufträge auf den einzelnen Maschinen als letzter bearbeitet werden kann, müssen jeweils von jeder Bearbeitung ein Pfeil zum Endpunkt $B_J$ ($J \in \{1,\ldots,3\}$) hinführen.

## 4.2. Die Benderssche Partitionsmethode zur Lösung gemischt-ganzzahliger Entscheidungsmodelle

BENDERS schlug in [9] eine Methode zur Lösung gemischt-ganzzahliger Optimierungsprobleme vor. Beispielsweise gaben FLEISCHMANN [19], LEMKE/SPIELBERG [40] und ZOUTENDIJK [58] Modifikationen an, um lineare gemischt-ganzzahlige Entscheidungsmodelle mit Binärvariablen lösen. BALAS [6] gab als erster Autor eine Anwendung der Bendersschen Partitionsmodelle auf Ablaufplanungsprobleme mit der Zielsetzung „Minimierung der Zykluszeit" an. Dieser Ansatz von BALAS wird in den folgenden Kapiteln ausgedehnt und auf andere Zielsetzungen übertragen. Zunächst soll die Benderssche Methode allgemein erläutert und ihre numerischen Eigenschaften untersucht werden.

Gegeben sei das gemischt ganzzahlige Programm

$$
\begin{aligned}
(4\text{-}17)\quad & \min \quad c^T * x + d^T * y \\
& \text{u.d.N.} \quad A * x + B * y \geqq b \\
& \qquad\qquad\qquad x \geqq 0,\ y \in \{0,1\},
\end{aligned}
$$

wobei A eine (m,n)-Matrix, B eine (m,n)-Matrix, $c,d \in \mathbb{R}^n$, $x \in \mathbb{R}_+^n$, $b \in \mathbb{R}^m$ und $y \in \{0,1\}^n$.

Wird y fixiert, so erhält man

$$
\begin{aligned}
(4\text{-}18)\quad & \min \quad c^T * x + \text{const.} \\
& \text{u.d.N.} \quad A * x \geqq b - B * y \\
& \qquad\qquad x \geqq 0.
\end{aligned}
$$

Das zu (4-18) duale Problem lautet:

$$
\begin{aligned}
(4\text{-}19)\quad & \max \quad u^T * (b - B * y) \\
& \text{u.d.N.} \quad A^T * u \leqq c \\
& \qquad\qquad u \geqq 0.
\end{aligned}
$$

Definition 4.1:

$$U := \{u \mid A^T * u \leq c,\ u \geq 0\}$$

U ist unabhängig von der (0,1) Variable y. Die Zielfunktion nimmt ihr Maximum mindestens in einem Eckpunkt $u^p$ an oder wächst unbeschränkt entlang eines Extremalstrahls. U besitzt endlich viele Extrempunkte und endlich viele Extremalstrahlen. Die Extremalstrahlen von U können durch die Bestimmung aller Extremalstrahlen des „homogenen" Polyeders $A^T * u \leq 0$, $u \geq 0$ bestimmt werden. Sie seien mit $v^s$ $(s = 1,\ldots,S)$ bezeichnet. Gilt für ein fixiertes y, daß ein Index s existiert, so daß $v^{sT} * (b - B*y) > 0$, so ist das Maximum von $u^T * (b - B*y)$ unbeschränkt. Dies bedeutet, daß Problem (4-18) keine zulässige Lösung besitzt. Im Hinblick auf das Ablaufplanungsproblem heißt dies, daß der Ablaufgraph F einen Zyklus enthält.

Da der Polyeder U nur endlich viele Extrempunkte $u^r$ $(r = 1,\ldots,R)$ und endlich viele Extremalstrahlen $v^s$ $(s = 1,\ldots,S)$ besitzt, kann (4-19) so geschrieben werden:

$$\begin{aligned} (4\text{-}20) \qquad & \max_{r=1,\ldots,R} \; u^{rT} * (b - B*y) \\ & \text{u.d.N.} \; v^{sT} * (b - B*y) \leq 0 \quad (s = 1,\ldots,S) \\ & \qquad\qquad u \geq 0. \end{aligned}$$

(4-17) kann dann so geschrieben werden:

$$\begin{aligned} (4\text{-}21) \qquad & \min_{y} \; (d^T * y + \max_{r=1,\ldots,R} \; u^{rT} * (b - B*y)) \\ & \text{u.d.N.} \; v^{sT} * (b - B*y) \leq 0 \quad (s = 1,\ldots,S) \\ & \qquad\qquad y \in \{0,1\}. \end{aligned}$$

Algorithmus 8:

*Name:* Verfahren von BENDERS [9] zur gemischt ganzzahligen Optimierung.

*Voraussetzung:* Gegeben sei c,d,A,B,m,n.

*Initialisierung:* Wähle einen beliebigen Vektor $y \in \{0,1\}^n$. Setze die obere Schranke $z^u = +\infty$, die untere Schranke $z^\ell = -\infty$.

*Iteration:*

1. Löse (4-19). Das Ergebnis ist ein Extrempunkt $u^r$ oder ein Extremalstrahl $v^s$.
2. Falls ein Extrempunkt gefunden wird, berechne

$$z_{IT} = d^T * y + u^r * (b - B * y).$$

Gilt $z_{IT} < z^u$, dann setze $z^u = z_{IT}$.

3. Löse das „ganzzahlige" Programm

$$\begin{aligned} \min \quad & z \\ \text{u.d.N.} \quad & z \geqq d^T * y + u^{rT} * (b - B * y) \quad (r = 1, \ldots, R^*) \\ & 0 \geqq v^{sT} * (b - B * y) \quad (s = 1, \ldots, S^*) \\ & y \in \{0,1\}, \end{aligned}$$

wobei $R^*, S^*$ die Zahl der gefundenen Extrempunkte bzw. Extremalstrahlen sind.

4. Setze $z^\ell = z$.

*STOPKRITERIUM:* Ist $z^\ell < z^u$. Wenn ja, wiederhole Schritt 1. bis 4. Wenn nein, löse mit dem Wert von y das Programm (4-18), das den kleinsten Wert von $z^u$ geliefert hat. Das optimale x,y ist gefunden.

Bemerkung 4.6:

Eigenschaften des Bendersschen Verfahrens (vgl. z.B. auch SALKIN [53]).

1. Eigenschaft: Eine optimale Lösung y aus Schritt 3 wiederholt sich nie.

2. Eigenschaft: In jeder Iterationsstufe wird ein neuer Extrempunkt oder Extremalstrahl gefunden.

3. Eigenschaft: Es ist jederzeit im Bendersschen Algorithmus eine untere Schranke und eine obere Schranke für den optimalen Zielfunktionswert von (4-17) vorhanden. Die unteren Schranken $z^{\ell}$ steigen dabei monoton (dies folgt daraus, daß das ganzzahlige Programm immer um eine Restriktion erweitert wird), während die obere Schranke $z^{u}$ nicht monoton zu fallen braucht.

Graphische Darstellung:

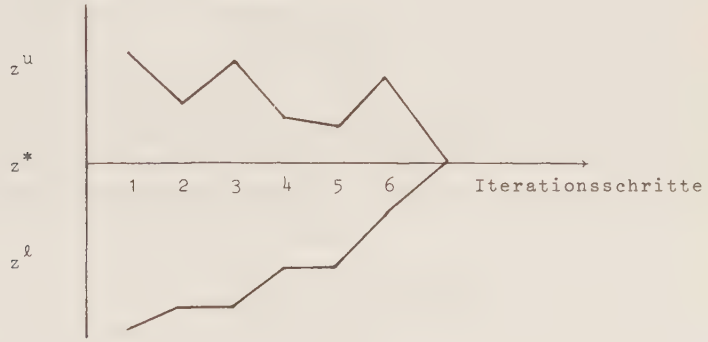



Abbildung 15

4. Eigenschaft: Die Effizienz des Bendersschen Verfahrens hängt von vier Faktoren ab:

i. vom Startpunkt $y \in \{0,1\}$,
ii. von der Anzahl der durchzuführenden Iterationen,
iii. von der Struktur des dualen Programms (4-18) und
iv. von der Lösungsmethode für das „rein" ganzzahlige Programm.

Es ist allgemein nicht möglich, für die Benderssche Methode die Zeitkomplexität abzuschätzen. Im Schritt 1. von Algorithmus 8 wird ein lineares Programm gelöst. Bis vor kurzem war nicht bekannt, ob lineare Programme zur Klasse $\mathcal{P}$ gehören. Erst durch eine Arbeit von

KACHIJAN [59] konnte allgemein gezeigt werden, daß lineare Optimierungsprobleme zur Klasse $\mathcal{P}$ gehören. Das in Schritt 3. zu lösende Programm ist bis auf die stetige Variable z ein rein ganzzahliges Programmierungsproblem, das zur Klasse der $\mathcal{NP}$-vollständigen Probleme gehört.

## 4.3. Die Anwendung der Bendersschen Partitionsmethode auf Ablaufplanungsprobleme

In diesem Abschnitt soll zuerst die Zielsetzung „Minimierung der Zykluszeit" untersucht werden, da sich die Aussagen für die anderen Zielsetzungen aus der Minimierung der Zykluszeit herleiten bzw. verallgemeinern lassen.

Gegeben sei das MANNE-Modell (4-12):

$$
\begin{aligned}
&\min \quad Z \\
&\text{u.d.N.} \quad h_{I[J]} + p_{I[J]} \leqq h_{I[J+1]} \qquad (I = 1,\ldots,N;\ J = 1,\ldots,M-1) \\
&\qquad h_{IJ} + p_{IJ} \leqq h_{I+L,J} + UB*(1-\delta_{I,L,J}) \\
&\qquad h_{I+L,J} + p_{I+L,J} \leqq h_{IJ} + UB*\delta_{I,L,J} \\
&\qquad (J = 1,\ldots,M;\ I = 1,\ldots,N-1;\ L = 1,\ldots,N-I) \\
&\qquad h_{I[M]} + p_{I[M]} \leqq Z \qquad (I = 1,\ldots,N) \\
&\qquad Z,\ h_{IJ} \geqq 0 \qquad (I = 1,\ldots,N;\ J = 1,\ldots,M) \\
&\qquad \delta_{I,L,J} \in \{0,1\} \qquad (J = 1,\ldots,M;\ I = 1,\ldots,N-1;\ L = 1,\ldots,N-I).
\end{aligned}
$$

Angenommen, es sei ein Vektor $\delta_{I,L,J}$ $(J = 1,\ldots,M;\ I = 1,\ldots,N-1;\ L = 1,\ldots,N-I)$ bekannt, der einen zulässigen Ablaufgraphen F beschreibt, dann stellt Modell (4-12) ein Netzwerk dar, d.h. das lineare Programm, das

durch die Wahl des Vektors $\delta_{I,L,J}$ entsteht, läßt sich in einem Netzwerk darstellen. Die Knoten dieses Netzwerks repräsentieren die Starttermine $h_{IJ}$ (I = 1,...,N; J = 1,...,M), während die Bewertungen der Pfeile die Bearbeitungsdauern darstellen. Gesucht ist in diesem Netzwerk derjenige Weg, der die Zeit vom Beginn der Bearbeitungen bis zum Ende der letzten Bearbeitung eines Auftrags minimiert. Ein solcher Weg in Netzwerken heißt auch *längster* Weg oder auch *kritischer* Weg in Netzwerken. Als *kritische* Dauer dieses Netzwerks sei der Termin bezeichnet, der den minimalen Zeitbedarf angibt, um alle Bearbeitungen in der festgelegten Reihe durchzuführen. Um den kritischen Weg zu ermitteln, der dann eine Lösung für das lineare Programm darstellen wird, müssen zunächst die frühestmöglichen Anfangstermine der N*M-Bearbeitungen (Knoten) ermittelt werden (vgl. z.B. auch BRUCKER/DINKELBACH [11], SUCHOWIZKI/RADTSCHIK [56]). Die Ermittlung dieser frühestmöglichen Beginnzeitpunkte kann dadurch geschehen, indem längste Wege vom Startknoten A zum Endknoten E ermittelt werden. Dies kann beispielsweise mit dem Algorithmus von FORD (vgl. z.B. [11] oder DIJKSTRA [17]) erreicht werden. Allerdings kann dies auch mit einer Modifikation von Algorithmus 3 zur Erzeugung semiaktiver Ablaufpläne berechnet werden. Im allgemeinen muß mit dem Verfahren von MARIMONT getestet werden, ob ein Vektor $\delta_{I,L,J}$ (J = 1,...,M; I = 1,...,N-1; L = 1,...,N-I) in Verbindung mit den technologischen Reihenfolgebedingungen einen zulässigen Ablaufplan darstellt. Ist dieser Plan zulässig, so kann mit Hilfe der in Schritt 1. ermittelten Zeilensummen die Reihenfolgen *FNM* auf den einzelnen Maschinen angegeben werden. Da die Reihenfolgen bekannt sind, kann also sofort der zugehörige semiaktive Ablaufplan angegeben werden, indem das Maximum des frühesten Verfügbarkeitstermins *HANM* und des Verfügbarkeitstermins der Maschine gebildet werden.

Ist der längste Weg von Knoten A nach Knoten E bestimmt, müssen noch die spätestmöglichen Beginnzeitpunkte der N*M Bearbeitungen ermittelt werden, ohne daß sich die Länge des ermittelten minimalen längsten Weges ändert. Dies kann durch die Bestimmung kürzester Wege vom Knoten E nach Knoten A geschehen. Es existieren zahlreiche Algorithmen zur Bestimmung kürzester Wege in Graphen. Um die spezielle Struktur eines Ablaufplanungsproblems auszunutzen, wird eine Spezialisierung eines Algorithmus zur Ermittlung kürzester Wege in einem Netzwerk vorgenommen.

Algorithmus 9:

*Name:* Verfahren zur Ermittlung spätester Starttermine *SHNM*(IJ) bei gegebener Dauer des längsten Weges in einem Ablaufplanungsnetzwerk.

*Voraussetzung:* Gegeben seien die Mengen *NM*, *CNM*, *PNM*, *HANM*, *FNM*, *HNM*, *DNM*.

*Iteration:*

1. Sortiere die Indizes aus *NM* in einer Indexmenge *SNM* so an, daß die Starttermine $h_{IJ}$ monoton fallend sind. Bezeichne das Maximum aller Endtermine von *DNM* mit *DNM**.
2. L = 1.
3. Wähle den L-ten Index aus *SNM*. Dies sei Index IJ.
4. Besitzt IJ weder eine technologische Nachfolgeoperation, noch einen Nachfolger auf Maschine J? Wenn nein, gehe zu Schritt 5. Wenn ja, ist IJ die letzte Bearbeitung aller Aufträge? Wenn ja, dann *SHNM*(IJ) = *HNM*(IJ). Gehe zu Schritt 8. Wenn nein, dann *SHNM*(IJ) = max {*DNM*(IJ), *DNM**-1} - *PNM*(IJ). Gehe zu Schritt 8.
5. Besitzt IJ mit IH eine technologische Nachfolgeoperation, aber keinen Maschinennachfolger? Wenn ja,

*SHNM*(IJ) = *SHNM*(IH) - *PNM*(IJ). Gehe zu Schritt 8. Wenn nein, gehe zu Schritt 6.
6. Besitzt IJ eine Maschinennachfolgeoperation KJ, aber keine technologische Nachfolgeoperation IH? Wenn ja, *SHNM*(IJ) = *SHNM*(KJ) - *PNM*(IJ). Gehe zu Schritt 8. Wenn nein, gehe zu Schritt 7.
7. *SHNM*(IJ) = MIN {*SHNM*(KJ), *PNM*(IH)} - *PNM*(IJ).
8. L = L + 1.

*STOPKRITERIUM:* Falls L > N*M, dann STOP. Wenn nein, gehe zu Schritt 3.

Satz 4.1:
Algorithmus 9 benötigt zur Konstruktion von kürzesten Wegen im Ablaufplanungsnetzwerk von Knoten E nach Knoten A $O(N^2*M^2)$.

Beweis: Der Sortierprozeß in Schritt 1. hat die Zeitkomplexität $O(N^2*M^2)$. In Schritt 4 wird zuerst ein Vergleich der Form

$$CNM(IJ) = 0 \quad \text{und} \quad FNM(IJ) = N$$

vorgenommen. Ist dieser Vergleich wahr (dies kann genau N-mal der Fall sein), so wird die Abfrage erreicht, ob $DNM(IJ) = DNM^*$ ist. Dies kann N-mal zutreffen, d.h., daß alle Aufträge unter Umständen zur gleichen Zeit und auf unterschiedlichen Maschinen fertiggestellt werden können. In Abhängigkeit von dieser Abfrage wird *SHNM* ein Wert zugewiesen.

Die einmalige Ausführung von Schritt 4. benötigt maximal drei Vergleiche. Da Schritt 4. insgesamt aber N*M-mal durchgeführt wird und der dritte Vergleich maximal N-mal ausgeführt wird, beträgt der Rechenaufwand $O(N^2*M)$.

In Schritt 5. wird ein Vergleich der Form

$$CNM(IJ) = 0 \quad \text{und} \quad FNM(IJ) \neq N$$

vorgenommen. Ist dieser Vergleich erfüllt (dies kann N-mal eintreten), erfolgt die Wertzuweisung an *SHNM*(IJ). Da Schritt 5. maximal N*M-mal ausgeführt wird, ist die Zeitkomplexität $O(N^2*M)$. In Schritt 6. erfolgt ein ähnlicher Vergleich wie in Schritt 5. Daher beträgt die Zeitkomplexität $O(N^2*M)$. Schritt 7. wird maximal N*M-mal ausgeführt, d.h., der Rechenaufwand ist O(N*M).

Da im Algorithmus 9 der Sortierprozeß die höchste Zeitkomplexität hat, beträgt der Rechenaufwand $O(N^2*M^2)$ für Algorithmus 9.

**Beispiel 4.4:**
Für den in Kapitel 3 in Abbildung 3 dargestellten Ablaufplan sollen früheste Beginnzeitpunkte und spätestmögliche Beginnzeitpunkte ermittelt werden.
Es ergibt sich mit Hilfe der Algorithmen 3 und 9 folgende Termine:

| NM | HNM | SHNM |
|---|---|---|
| 11 | 0 | 6 |
| 21 | 15 | 19 |
| 31 | 4 | 10 |
| 12 | 16 | 20 |
| 22 | 5 | 5 |
| 32 | 12 | 12 |
| 13 | 22 | 22 |
| 23 | 0 | 0 |
| 33 | 16 | 16 |

Kritische Ereignisse sind durch den gleichen frühesten Startzeitpunkt und spätestmöglichen Beginnzeitpunkt gekennzeichnet. Hier wären also die Operationen 22, 32, 13, 23 und 33 kritische Operationen. Den kritischen Weg von Knoten A zu Knoten E erhält man dadurch, daß die Operationen aufsteigend nach Beginnzeitpunkten sortiert werden. Sehr oft ergeben sich mehrere kritische Wege. Diese können nach dem Sortierprozeß dadurch erkannt werden, daß zwei Operationen in der sortierten Menge keine Verbindung zueinander haben. Dann kann man durch Suche nach ihrem Vorgänger in der sortierten Menge die kritischen Wege ermitteln. In Beispiel 4.4 lautet der kritische Weg {(A,23),(23,22),(22,32),(32,33),(33,13),(13,E)} Die Länge des kritischen Weges beträgt 23 ZE.

Graphische Darstellung:

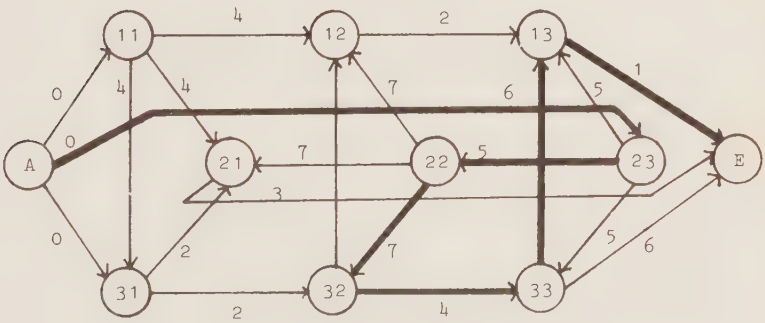



Abbildung 16
(Der kritische Weg ist dick eingezeichnet)

Die Lösung des linearen Programms bei Vorgabe eines Vektors $\delta_{I,L,J}$, der einen zulässigen Ablaufplan beschreibt, ist auf das Bestimmen eines kritischen Weges in einem Netzwerk zurückgeführt worden. Die Struktur des linearen Programms soll noch weiter studiert werden.

Werden in dem linearen Programm, das auf (4-12) zurückgeht, die Restriktionen zuerst nach technologischen Reihenfolgebedingungen, Zyklusbeschränkungen und dann nach Auftragsfolgebedingungen angeordnet und die Starttermine $h_{IJ}$ nach Maschinen gruppiert, so ergibt sich für das lineare Programm folgende Struktur:

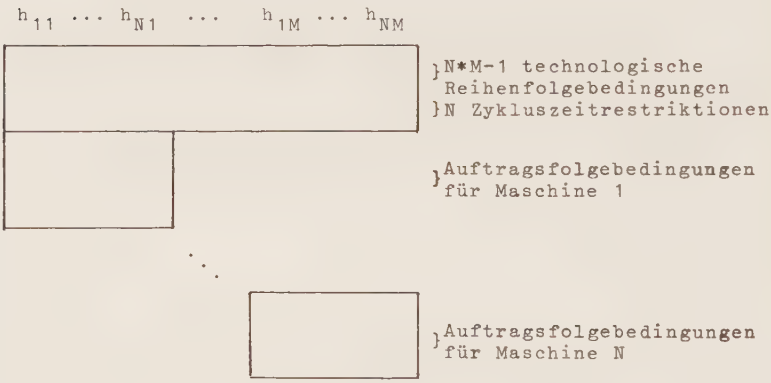



Abbildung 17

Dieses lineare Programm besitzt eine „blockangulare" Struktur, auf die sich zur Lösung auch Dekompositionsalgorithmen anwenden ließen. Interessanter ist jedoch die Struktur der Nebenbedingungen zu untersuchen, um Rückschlüsse auf die zu einer primalen Lösung gehörende Dualvariable zu erhalten, die in der Bendersschen Methode benötigt werden.

Zunächst soll die Koeffizientenmatrix des dualen linearen Programms untersucht werden. Dazu sollen zunächst einige Bezeichnungen eingeführt werden.

Definition 4.2:

$I_1$ sei die Indexmenge, die alle Pfeile enthält, die von A ausgehen: $I_1 := \{(A,IJ) \mid A \text{ - Quelle des Netzwerkes}\}$.

$I_2$ sei die Indexmenge, die die technologischen Reihenfolgen und die Auftragsfolgen beschreibt:

$$I_2 := \{(IJ,KL) \mid (IJ,KL) \in C \cup D^*\}.$$

$I_3$ sei die Indexmenge, die die Pfeile der letzten Bearbeitungen zur Senke E enthält:

$$I_3 := \{(IJ,E) \mid E \text{ - Senke des Netzwerkes}\}.$$

Mit $h_A$ sei der Beginnzeitpunkt der Planungsperiode bezeichnet, mit $h_E$ sei der gesuchte Endzeitpunkt bezeichnet, mit $p_A$ sei die Dauer von A bezeichnet. Nach Vorgabe eines Vektors $\delta_{I,L,J}$, der einen zulässigen Ablaufplan beschreiben möge, läßt sich das lineare Programm mit den eingeführten Bezeichnungen schreiben.

$$\begin{array}{lll} (4\text{-}22) & \min & -h_A + h_E \\ & \text{u.d.N.} & -h_A + h_{IJ} \geq p_A \quad \forall\ (A,IJ) \in I_1 \\ & & -h_{IJ} + h_{KL} \geq p_{IJ} \quad \forall\ (IJ,KL) \in I_2 \\ & & -h_{IJ} + h_E \geq p_{IJ} \quad \forall\ (IJ,E) \in I_3 \end{array}$$

In der Formulierung (4-22) sind die Beginnzeitpunkte vorzeichenunbeschränkt. Setzt man z.B. $h_A$ und $p_A$ gleich Null, so sind die Variablen $h_{11},\ldots,h_{NM}$ alle nichtnegativ. Die Zielfunktion von (4-22) ist nichts anderes als der minimale längste Weg in dem Netzwerk F.

Definition 4.3:

Mit $u_{A,IJ}$ seien die Dualvariablen bezeichnet, die zu den Restriktionen aus der Indexmenge $I_1$ korrespondieren. Entsprechend $v_{IJ,KL}$ für die Indexmenge $I_2$ und $w_{IJ,E}$ für die Indexmenge $I_3$.

Dann lautet das zu (4-22) gehörende duale lineare Programm:

$$(4\text{-}23) \quad \max \sum_{(A,IJ)\in I_1} p_A * u_{A,IJ} + \sum_{(IJ,KL)\in I_2} p_{IJ} * v_{IJ,KL} + \sum_{(IJ,E)\in I_3} p_{IJ} * w_{IJ,E}$$

$$\text{u.d.N.} \quad -\sum_{(A,IJ)\in I_1} u_{A,IJ} = -1$$

$$u_{A,IJ} - w_{IJ,E} + \sum_{KL\in\Gamma(IJ)} - v_{IJ,KL} + \sum_{IJ\in\Gamma^{-1}(KL)} v_{IJ,KL} = 0 \quad \forall \; IJ$$

$$\sum_{(IJ,E)\in I_3} w_{IJ,E} = 1$$

$$u_{A,IJ},\ v_{IJ,KL},\ w_{IJ,E} \geqq 0$$

Die Anzahl der Gleichungen in (4-23) beträgt N*M + 2.
$\Gamma$ ist die Nachfolgermenge (oder die Pfeile, die von IJ nach KL gehen) von Knoten IJ. $\Gamma^{-1}$ ist die inverse Nachfolgemenge.

**Definition 4.4:**

(Vgl. auch HU [30, S. 148 f.])
Die Beschränkungen eines linearen Programms seien $A*x = b$, $x \geqq 0$, dabei sei A eine ganzzahlige Matrix mit linear unabhängigen Zeilen, b sei ein ganzzahliger Vektor. Die Matrix A heißt unimodular genau dann, wenn eine der drei zueinander äquivalenten Bedingungen erfüllt sind.

Bedingung 1: Die Determinante jeder Basis B von A ist +1 oder -1.

Bedingung 2: Die Ecken des konvexen Polyeders C, der durch $A*x = b$, $x \geqq 0$ gebildet wird, sind für alle ganzzahligen Vektoren b ganzzahlig.

Bedingung 3: Die Inverse $B^{-1}$ jeder Basis B ist eine ganzzahlige Basis.

Um zu überprüfen, ob eine Matrix unimodular ist, ist die Definition 4.3 nicht geeignet, da man z.B. alle möglichen Basen B von A untersuchen müßte. VEINOTT und DANTZIG gaben in [57] einen Satz an, mit dem man konstruktiv die Unimodularität einer Matrix überprüfen kann.

Satz 4.2: (VEINOTT/DANTZIG [57])
Eine Matrix A ist unimodular, wenn die folgenden Bedingungen erfüllt sind:

Bedingung $1^*$ : Jede Spalte von A enthält höchstens zwei von Null verschiedene Koeffizienten.

Bedingung $2^*$ : Jeder Koeffizient von A ist 0, +1 oder -1. Die Matrix A kann in zwei disjunkte Zeilenmengen $R_1$ und $R_2$ eingeteilt werden, und zwar so, daß

$2^*a$: in einer Spalte zwei von Null verschiedene Koeffizienten mit gleichem Vorzeichen nicht in der gleichen Zeilenmenge enthalten sind,

$2^*b$: in einer Spalte zwei von Null verschiedene Koeffizienten mit verschiedenem Vorzeichen in der gleichen Zeilenmenge enthalten sind.

Bemerkung 4.7:
Satz 4.2 gilt auch, falls $R_1$ oder $R_2$ gleich der leeren Menge ist. Hat man von der Koeffizientenmatrix A gezeigt, daß sie unimodular ist, so ist auch $A^T$ unimodular (vgl. z.B. HU [30, S. 151]).

Satz 4.3:
Die Koeffizientenmatrix des linearen Programms (4-23) ist unimodular.

Beweis: Jede Spalte der Koeffizientenmatrix von (4-23) besitzt die Koeffizienten -1, 0, 1; -1 und 1 kommen genau einmal vor. Diese beiden Koeffizienten entsprechen

jeweils einer Restriktion in (4-22). Setzt man $R_1$ gleich der leeren Menge und $R_2$ gleich der gesamten Zeilenmenge von (4-23), so sind die Bedingungen 2*a und 2*b erfüllt, d.h., die Koeffizientenmatrix von (4-23) ist unimodular. Mit Bemerkung 4.1 folgt auch sofort die Unimodularität der Matrix der Koeffizienten des linearen Programms (4-22).

Bemerkung 4.8:
Satz 4.3 garantiert bei Ganzzahligkeit der Bearbeitungsdauern die Ganzzahligkeit der Beginnzeitpunkte $h_{IJ}$. Außerdem gehört zu einer optimalen ganzzahligen Lösung von (4-22) immer eine rein ganzzahlige optimale Lösung des dualen linearen Programms (4-23). Der folgende Satz charakterisiert die Werte der Dualvariablen.

Satz 4.4:
Jede Basislösung von (4-23) besitzt die Werte Null oder Eins.

Beweis: Gegeben sei eine Ecke des konvexen Polyeders, der durch die Restriktionen von (4-23) beschrieben wird. Die zu dieser Ecke gehörende Basismatrix sei mit B bezeichnet. Die Komponenten des Vektors $(u_{A,IJ}, v_{IJ,KL}, w_{IJ,E})$, die den Basisvariablen entsprechen, seien mit x bezeichnet. Dann gilt

$$(4\text{-}24) \qquad B*x = \begin{pmatrix} -1 \\ 0 \\ \vdots \\ 1 \end{pmatrix} \quad \text{mit} \quad x \geq 0.$$

Nach der Cramerschen Regel für lineare Gleichungssysteme gilt

$$x_i = \frac{|B_i|}{|B|},$$

wobei $|B|$ die Determinante der Matrix ist und $|B_i|$ die Determinante derjenigen Matrix, die durch Streichen der

i-ten Spalte in der Matrix B entsteht und um die rechte Seite von (4-24) ergänzt wird.

Da B Basis und unimodular ist, gilt mit Bedingung 1 von Definition 4.3:

$$|B| = +1 \quad \text{oder} \quad -1.$$

Es muß noch $|B_i|$ untersucht werden:

Die i-te Spalte der Matrix B sei durch $\begin{pmatrix} -1 \\ 0 \\ \vdots \\ 1 \end{pmatrix}$ ersetzt worden.

Falls diese Spalte von einer anderen Spalte linear abhängig ist, gilt: $|B_i| = 0$. Ist die i-te Spalte von den übrigen Spalten linear unabhängig, so folgt mit Hilfe des Austauschsatzes von STEINITZ (vgl. z.B. KOWALSKY [36, S. 37]), daß $B_i$ Basis ist, d.h. $|B_i| = +1$ oder $-1$. Zusammen mit der Vorzeichenbeschränkung $x \geq 0$ ergibt sich also $x_i = 0$ oder 1 für alle Indizes i.

**Satz 4.5:**
Für das Netzwerk F, das dem linearen Programm (4-22) entspricht, sei ein kritischer Weg $\langle(A,IJ),(IJ,KL),\ldots,(OP,E)\rangle$ bekannt. Dann besitzen die Dualvariablen $u_{A,IJ}$, $v_{IJ,KL}$, $\ldots$, $w_{OP,E}$ den Wert Eins. Alle übrigen Dualvariablen besitzen den Wert Null.

**Beweis:** Die Restriktionen in (4-22), die den kritischen Weg $\langle(A,IJ),\ldots,(OP,E)\rangle$ im Netzwerk F bilden, sind als Gleichungen erfüllt, während diejenigen Restriktionen, die nicht auf dem kritischen Weg liegen, als strikte Ungleichungen erfüllt sind. Daraus ergibt sich nach dem Satz des komplementären Schlupfs für duale lineare Programme, daß die zugehörigen Dualvariablen den Wert Null haben.

Aus (4-22) seien alle Restriktionen entfernt, die nicht auf dem kritischen Weg liegen. Das kritische Ereignis IJ (IJ ≠ A, IJ ≠ E) kommt dann in den Restriktionen (4-22) genau zweimal vor, einmal mit positivem Vorzeichen und einmal mit negativem Vorzeichen. Dies bedeutet für das duale Programm (4-23), daß die Restriktionen folgendes Aussehen haben:

$$
\begin{aligned}
u_{A,IJ} &= 1 \quad \text{(IJ fix)} \\
u_{A,IJ} - v_{IJ,KL} &= 0 \\
&\vdots \\
v_{RT,UV} - v_{OP,E} &= 0 \\
v_{OP,E} &= 1 \\
u_{A,IJ}, v_{IJ,KL}, \ldots, v_{RT,uv}, v_{OP,E} &\geq 0.
\end{aligned}
$$

Aus dieser Struktur ergibt sich sofort, daß die Dualvariablen den Wert Eins besitzen.

Bemerkung 4.9:
Satz 4.5 gilt auch, falls mehr als ein Weg existiert. Dann kommt ein kritisches Ereignis mehrfach in den dualen Gleichungen von (4-23) vor. Aus der Forderung, daß die dualen Variablen ganzzahlig sind, ergibt sich sofort die Behauptung.

Satz 4.2 bis 4.5 erlaubt es, aus einer primalen Lösung des Problems (4-22), das ein kritisches Wegproblem ist, sofort die Lösung des dualen Programms (4-23) anzugeben. Dies ist für die Anwendung der Bendersschen Partitionsmethode wichtig. Um mit dem Bendersschen Algorithmus starten zu können, wird ein Vektor mit Null-Eins Variablen benötigt. Satz 4.6 gibt Auskunft über Startvektoren, die zulässige Ablaufpläne repräsentieren.

**Satz 4.6:**
Die Vektoren $\delta_{I,L,J} = (0,\ldots,0)^T$ und $\delta_{I,L,J} = (1,\ldots,1)^T$ stellen für das Ablaufplanungsproblem (4-12) immer zulässige Reihenfolgen dar.

**Beweis:** Der Nullvektor erzeugt auf Maschine J (J = 1,...,M) die Auftragsfolge NJ,...,1J, d.h., jede Operation wird eindeutig angeordnet und damit ist das Netzwerk F ohne Zyklen. Der Eins-Vektor erzeugt auf Maschine J (J = 1,...,M) die Auftragsfolge 1J,2J,...,NJ. Das Netzwerk F ist also ohne Zyklen, d.h., es stellt einen zulässigen Ablaufplan dar.

Im Laufe der Iteration von Algorithmus 8 kann jedoch ein Vektor $\delta_{I,L,J}$ erzeugt werden, der keinen zulässigen Ablaufplan darstellt, d.h., das Problem (4-22) besitzt keine zulässige Lösung. Aus der Dualitätstheorie der linearen Programmierung folgt dann, daß das duale Programm (4-23) entlang eines Extremalstrahls unbeschränkt wächst, oder das duale Programm besitzt keine Lösung. Hier existiert jedoch für die spezielle Struktur des dualen Programms mindestens eine zulässige Lösung. Wird in (4-22) die Restriktionenmenge um Vorzeichenbeschränkungen $h_{IJ} \geq 0 \;\forall\; IJ \in I_1 \cup I_2 \cup I_3$ erweitert, so ist das zu (4-22) gehörende duale lineare Programm ein spezielles lineares Maximumproblem, das mindestens die Lösung $(u_{A,IJ}, v_{IJ,KL}, w_{IJ,E}) = (0,\ldots,0)^T$ besitzt. Es muß also jetzt die Form des Extremstrahls gefunden werden, um im Bendersschen Programm weiterarbeiten zu können.

**Satz 4.7:**
Gegeben sei das lineare Programm (4-22) mit Vorzeichenbeschränkungen. Das zu dem linearen Programm korrespondierende Netzwerk (Graph) möge einen Zyklus der Form $<(IJ,KL),\ldots,(RS,IJ)>$ enthalten. Dann gilt für das zu (4-22) gehörende duale Programm, daß die Zielfunktion des Dualen entlang des Extremalstrahls $v_{IJ,KL} = \ldots = v_{RS,IJ} = 1$ unbeschränkt wächst.

Beweis: Da die Koeffizientenmatrix des zu (4-22) gehörenden dualen linearen Programms unimodular ist, verändern sich die Werte der Koeffizientenmatrix beim Pivotieren nicht, d.h., es kommen nur die Werte 0,1,-1 vor. Aus den Dualitätssätzen der linearen Programmierung weiß man, daß die Zielfunktion unbeschränkt wächst. Dies ist im Simplextableau daran zu erkennen, daß keine Pivotzeilen mehr bestimmt werden, da die in Betracht kommenden Elemente alle ein negatives Vorzeichen haben. Die Werte mit (-1) multipliziert, die in dieser Spalte stehen, geben die Richtung des Extremalstrahls an, d.h., hier besitzen sie den Wert 1.

Die in Schritt 3. von Algorithmus 8 zu bildende Restriktion für das „ganzzahlige" Unterprogramm soll näher untersucht werden. Die allgemeine Form lautet:

$$
\begin{array}{lll}
(4\text{-}24) & \min & z \\
& \text{u.d.N.} & z \geqq d^T * y + u^{rT} * (b - B * y) \qquad (r = 1,\ldots,R^*) \\
& & 0 \geqq v^{sT} * (b - B * y) \qquad (s = 1,\ldots,S^*) \\
& & y \in \{0,1\}.
\end{array}
$$

Falls der Vektor $\delta_{I,L,J}$ einen zulässigen Ablaufplan darstellt, wird in Schritt 2. ein Extrempunkt $u^r$ berechnet. Bezeichnet man mit $D^+$ die Pfeile der Auftragsreihenfolgen, die im Vektor $\delta_{I,L,J}$ einer Eins entsprechen, und mit $D^-$ die Pfeile der Auftragsreihenfolgen, die im Vektor $\delta_{I,L,J}$ einer Null entsprechen, mit CP (Critical Path) den kritischen Weg im Netzwerk F, mit $\ell_{CP}$ die Länge des kritischen Weges, mit $CP \cap D^+$ $(CP \cap D^-)$ sei

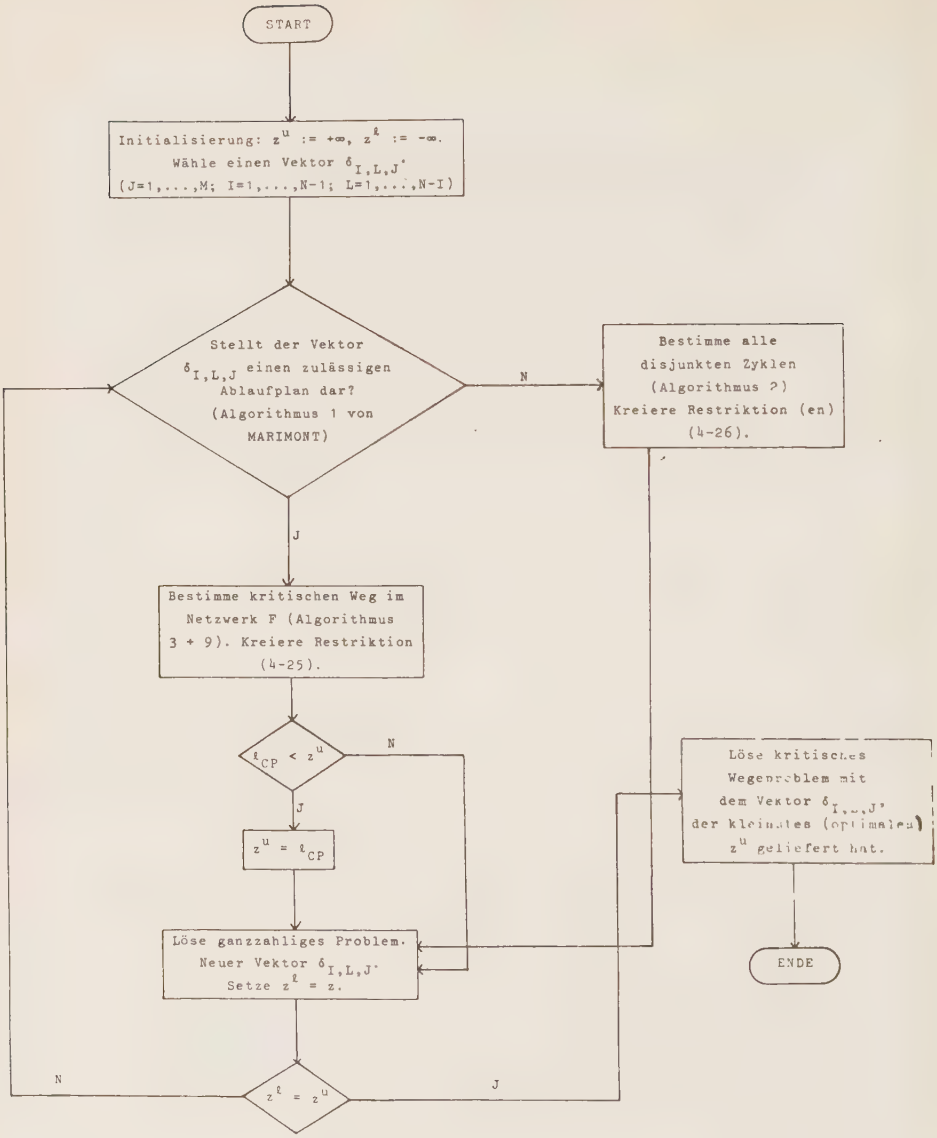
die Menge der Pfeile bezeichnet, die sowohl auf dem kritischen Weg liegen als auch zu Auftragsreihenfolgen gehören, die im Vektor $\delta_{I,L,J}$ einer Eins (Null) entsprechen und mit card $(CP \cap D^{+}_{-})$ sei die Anzahl der Pfeile bezeichnet, die in dieser Menge liegen. So erhält man für die ersten Restriktionen von (4-24) folgende Form:

$$(4\text{-}25) \quad z \geq \underbrace{\ell_{CP} - UB*\text{card}\,(CP \cap D^{+})}_{:= u^{rT}*b} + \underbrace{\sum_{CP\cap D^{+}} UB*\delta_{I,L,J}}_{:= u^{rT}*} \underbrace{- \sum_{CP\cap D^{-}} UB*\delta_{I,L,J}}_{B*y}$$

Stellt der in Schritt 2 von Algorithmus 8 gegebene Vektor $\delta_{I,L,J}$ keinen zulässigen Ablaufplan dar, d.h., es liegt ein Zyklus CY der Länge $\ell_{CY}$ vor, so erhält die Restriktion für (4-24) die Gestalt

$$(4\text{-}26) \quad 0 \geq \ell_{CY} - UB*\text{card}\,(CY \cap D^{+}) + \sum_{CY\cap D^{+}} UB*\delta_{I,L,J} - \sum_{CY\cap D^{-}} UB*\delta_{I,L,J}.$$

Das folgende Flußdiagramm gibt eine Spezialisierung der Bendersschen Partitionsmethode für Job-Shop-Probleme mit der Zielsetzung „Minimierung der Zykluszeit" an.




Spezialisierung der Bendersschen Partitionsmethode für Scheduling-Probleme mit der Zielfunktion „Minimierung der Zykluszeit"

**Beispiel 4.5:**

Für Beispiel 1.2 mit den Matrizen

$$P = \begin{pmatrix} 4 & 2 & 1 \\ 3 & 7 & 5 \\ 2 & 4 & 6 \end{pmatrix} \text{ und } Q = \begin{pmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \\ 1 & 2 & 3 \end{pmatrix}$$

soll mit Hilfe der Benderssschen Partitionsmethode derjenige Ablaufplan ermittelt werden, der die Zykluszeit minimiert. In Beispiel 4.2 ist das zugehörige gemischt ganzzahlige Optimierungsmodell dargestellt.

**Initialisierung von Algorithmus 8:**

Wahl eines beliebigen Vektors $\delta_{I,L,J}$ ($J = 1,\ldots,M$; $I = 1,\ldots,N-1$; $L = 1,\ldots,N-I$). Dies sei der Nullvektor, der ein zulässiger Ablaufplan ist. Das resultierende Netzwerk F für Beispiel 4.5, das ohne Pfeilbewertungen dargestellt ist, hat dann das Aussehen:

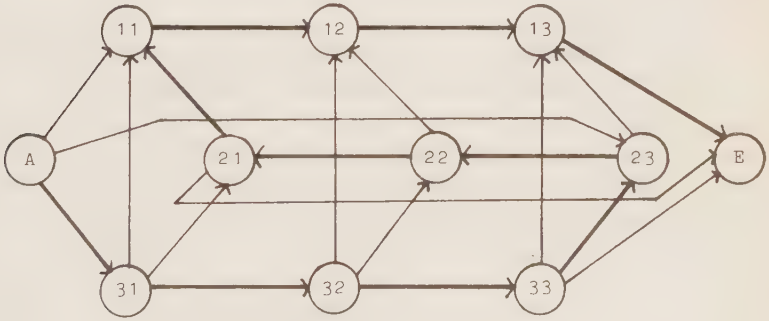



Abbildung 18

Schritt 1:
Die frühest möglichen Startzeitpunkte und spätest erlaubten Startzeitpunkte lauten:

| IJ | HNM | SHNM |
|---|---|---|
| 11 | 27 | 27 |
| 21 | 24 | 24 |
| 31 | 0 | 0 |
| | | |
| 12 | 31 | 31 |
| 22 | 17 | 17 |
| 32 | 2 | 2 |
| | | |
| 13 | 33 | 33 |
| 23 | 12 | 12 |
| 33 | 6 | 6 |

.

Alle Operationen sind kritische Operationen. Die Länge des kritischen Weges, der in Abbildung 18 dick eingezeichnet ist, beträgt 34 ZE.

Schritt 2:
Berechnung von $z_{IT} = 34$. Daraus ergibt sich $z^u = 34$. Es wird die folgende Restriktion generiert:

$$z + UB*\delta_{233} + UB*\delta_{121} \geq 34.$$

Das in Schritt 3 von Algorithmus 8 zu lösende „ganzzahlige" Programm lautet dann:

$$\min z$$
$$z \geq 34 - UB*\delta_{233} - UB*\delta_{121}.$$

Lösung dieses Programms ist $\delta_{233} = \delta_{121} = 1$. Daraus ergibt sich für die untere Schranke $z^{\ell} = 34 - 2*UB$, d.h., die Lösung ist noch nicht optimal. Mit dem Vektor

$\delta_{I,L,J} = (1,0,0,0,0,0,0,0,1)^T$ wird zu Schritt 1 von Algorithmus 8 zurückgegangen. Die folgenden Iterationen sind in einer Tabelle zusammengefaßt.

| Iterations-nummer | Restriktionen für das Benderssche Unterprogramm | | | | | | | | | | | Obere Schranke | Neuer Lösungsvektor | Untere Schranke |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | z | $\delta_{121}$ | $\delta_{131}$ | $\delta_{231}$ | $\delta_{122}$ | $\delta_{132}$ | $\delta_{232}$ | $\delta_{123}$ | $\delta_{133}$ | $\delta_{233}$ | $\geqq$ rechte Seite | $z^u$ | $\delta_{I,L,J}$ | $z^\ell$ |
| 0 | - | - | - | - | - | - | - | - | - | | - | $+\infty$ | $(0,0,0,0,0,0,0,0,0)^T$ | $-\infty$ |
| 1 | 1 | UB | | | | | | | | UB | 34 | 34 | $(1,0,0,0,0,0,0,0,1)^T$ | 34-2UB |
| 2a | 1 | | | | | | UB | | | | 16 | 16 | $(1,0,0,1,0,1,0,0,1)^T$ | 16-UB |
| 2b | 1 | | | | UB | | UB | | | | 16 | 16 | | |
| 3 | 0 | | | | UB | -UB | UB | | | | 13-2UB | - | $(1,0,0,0,0,1,0,0,1)^T$ | - |
| 4 | 1 | | | | | | -UB | | UB | | 23-UB | 16 | $(1,0,0,0,0,1,0,1,1)^T$ | 23-UB |
| 5 | 1 | | | | | UB | -UB | | -UB | | 25-2UB | 16 | $(1,0,0,0,1,1,0,1,1)^T$ | 25-UB |
| 6 | 1 | | | | UB | -UB | | | | | 24-UB | 16 | $(1,0,0,1,1,1,0,1,1)^T$ | 16 |

Der bezüglich der Zykluszeitminimierung optimale Ablaufplan ist in einem Ganttdiagramm dargestellt.

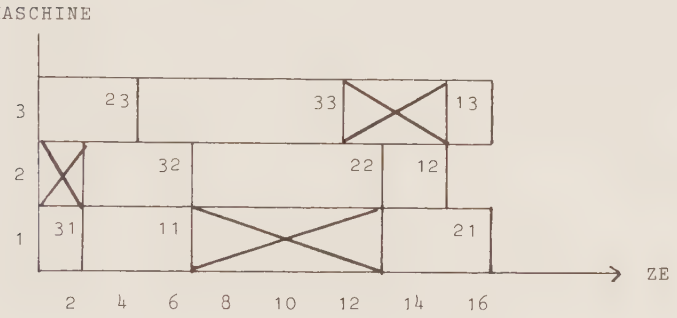



Abbildung 19

Nachdem die Anwendung auf Ablaufplanungsprobleme mit der Zielsetzung „Minimierung der Zykluszeit" untersucht wurde, soll die Spezialisierung auf die „Minimierung der Durchlaufzeit" vorgenommen werden.

Gegeben sei das MANNE-Modell

$$(4\text{-}8) \quad \min \sum_{I=1}^{N} (h_{I[M]} + p_{I[M]})$$

$$\text{u.d.N.} \quad h_{I[J]} + p_{I[J]} \leqq h_{I[J+1]} \qquad (I = 1,\ldots,N;\ J = 1,\ldots,M-1)$$

$$h_{IJ} + p_{IJ} \leqq h_{I+L,J} + UB*(1-\delta_{I,L,J})$$

$$h_{I+L,J} + p_{I+L,J} \leqq h_{IJ} + UB*\delta_{I,L,J}$$

$$(J = 1,\ldots,M;\ I = 1,\ldots,N-1;\ L = 1,\ldots,N-I)$$

$$h_{IJ} \geqq 0 \qquad (I = 1,\ldots,N;\ J = 1,\ldots,M)$$

$$\delta_{I,L,J} \in \{0,1\} \qquad (J = 1,\ldots,M;\ I = 1,\ldots,N-1;\ L = 1,\ldots,N-I)$$

Ist ein Vektor $\delta_{I,L,J}$ (J = 1,...,M; I = 1,...,N-1; L = 1,...,N-I) bekannt, der einen zulässigen Ablaufplan beschreibt, so geht das Modell (4-8) in einen Ablaufgraphen F über. Die Knoten dieses Graphen repräsentieren die Starttermine $h_{IJ}$ (I = 1,...,N; J = 1,...,M), während die Bewertungen der Pfeile die Bearbeitungsdauern darstellen. Ergänzt man den Graphen F um einen Startknoten A, von dem aus Pfeile mit der Bewertung Null zu den jeweils ersten Bearbeitungen der N Aufträge ausgehen und um N Endknoten, die von der jeweils letzten Bearbeitung der N Aufträge mit der zugehörigen Bearbeitungszeit als Bewertung ausgehen, so erhält man einen Graphen F, der gewisse Ähnlichkeiten mit dem linearen Programm (4-22) besitzt. (4-22) stellt allerdings ein Netzwerk dar, in dem der kritische Weg ermittelt werden soll. Der Graph F stellt kein Netzwerk dar, da in diesem Graphen N Senken existieren. F läßt sich so interpretieren, den jeweils längsten Weg vom Knoten A zu den N Endknoten zu suchen. Eine Bestimmung des kritischen Weges entfällt, da der kürzeste Weg von einem Endknoten zu Startknoten gleich dem längsten Weg vom Startknoten zum Endknoten ist, d.h., die Lösung des linearen Programms ist auf die Bestimmung der längsten Wege im Graphen F zurückgeführt. Wesentlich ist es im Bendersschen Programm, die zu einer primalen Lösung gehörenden Dualvariablen zu kennen. Bei der Minimierung der Zykluszeit hatten die zum kritischen Weg korrespondierenden Dualvariablen den Wert Eins. Dies trifft hier nicht mehr zu. Zerlegt man die Optimierungsaufgabe (4-8) in N Optimierungsaufgaben mit der Zielfunktion

$$
\begin{aligned}
\min \quad & h_{K[M]} + p_{K[M]} \\
\text{u.d.N.} \quad & h_{I[J]} + p_{I[J]} \leqq h_{I[J+1]} \qquad (I = 1,\ldots,N;\ J = 1,\ldots,M-1) \\
& h_{IJ} + p_{IJ} \leqq h_{I+L,J} + UB*(1-\delta_{I,L,J}) \\
& h_{I+L,J} + p_{I+L,J} \leqq h_{IJ} + UB*\delta_{I,L,J} \\
& (J = 1,\ldots,M;\ I = 1,\ldots,N-1;\ L = 1,\ldots,N-I) \\
& h_{IJ} \geqq 0 \qquad (I = 1,\ldots,N;\ J = 1,\ldots,M),
\end{aligned}
$$

wobei $K \in \{1,\ldots,N\}$, so liefert die Lösung dieser N Programmierungsmodelle bei festem $\delta_{I,L,J}$ jeweils den gesuchten Endtermin eines Auftrags K. Dieser Lösung ist ein Weg vom Knoten A nach Knoten $D_K$ zugeordnet, die denjenigen Restriktionen im linearen Programm entsprechen, die bindend sind. Mit Hilfe der Sätze 4.4, 4.5 und 4.6 kann dann geschlossen werden, daß die Dualvariablen den Wert Eins besitzen. Wird dieser Prozeß N-mal für K = 1,...,N durchgeführt, so lassen sich den Restriktionen von (4-8) jeweils N verschiedene Dualvariablen zuordnen. Die Summe dieser N Dualvariablen entspricht dann der gesuchten Dualvariablen für Modell (4-8) bei festem zulässigen $\delta_{I,L,J}$.

Ist im Ablaufgraphen F ein Zyklus, so läßt sich Satz 4.7 anwenden. Die im Schritt 3. von Algorithmus 8 zu bildende Restriktion für das „ganzzahlige" Unterprogramm soll näher untersucht werden.

Bezeichnet man mit $W(A \cap D_K)$ die Menge der Pfeile, die auf dem längsten Weg von A nach $D_K$ liegen (K fest), mit $(W(A \cap D_K) \cap D^+)$, die Menge der Pfeile, die sowohl auf dem längsten Weg von A nach $D_K$ liegen als auch zu Auftragsfolgebedingungen gehören, die durch die Wahl von $\delta_{I,L,J}$ auf Eins entstanden sind, mit card (IJ,LJ), die Häufigkeit, wie oft ein Pfeilpaar (IJ,JL) vorkömmt und mit $\ell_{W(A \cap D_K)}$ die Länge des längsten Weges von A nach $D_K$,

so kann die allgemeine Form dieser Restriktion angegeben werden.

$$(4\text{-}27)\quad z \geq \sum_{K=1}^{N} \ell_w(A \cap D_K) - UB^* \sum_{K=1}^{N} \sum_{(IJ,LJ)\in(W(A\cap D_K)\cap D^+)} \text{card}\,(IJ,LJ)$$

$$+ UB^* \sum_{K=1}^{N} \sum_{(IJ,LJ)\in(W(A\cap D_K)\cap D^+)} \delta_{I,L,J}$$

$$- UB^* \sum_{K=1}^{N} \sum_{(IJ,LJ)\in(W(A\cap D_K)\cap D^-)} \delta_{I,L,J}$$

Stellt der Vektor $\delta_{I,L,J}$ keinen zulässigen Ablaufplan dar, so muß im Bendersschen Unterprogramm die Restriktion (4-27) eingeführt werden.

Beispiel 4.6:

Für Beispiel 1.2 soll ein optimaler Ablaufplan bezüglich der Zielsetzung „Minimierung der Durchlaufzeit" bestimmt werden.

Setze $z^u = +\infty \qquad z^\ell = -\infty$

$\delta_{I,L,J} = (0,0,0,0,0,0,0,0,0)^T$.

Daraus resultiert der Ablaufgraph

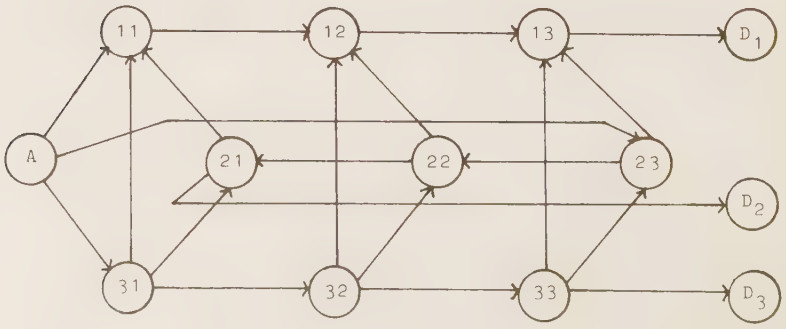



Abbildung 20

Die Bestimmung des längsten Weges von A nach $D_1$ ergibt:

$<(A,31),(31,32),(32,33),(33,23),(23,22),(22,21),(21,11),$
$(11,12),(12,13),(13,D_1)>.$

Die Bestimmung des längsten Weges von A nach $D_2$ ergibt:

$<(A,31),(31,32),(32,33),(32,23),(23,22),(22,21),(21,D_2)>.$

Die Bestimmung des längsten Weges von A nach $D_3$ ergibt:

$<(A,31),(31,32),(32,33),(33,D_3)>.$

Aufgrund der theoretischen Herleitung ergibt sich folgendes ganzzahlige Unterprogramm:

$$\min z$$
$$z \geq 73 - UB*\delta_{121} - 2*UB*\delta_{233}.$$

Lösung dieses Programms ist $\delta_{121} = 1$, $\delta_{233} = 1$. Die obere Schranke für die Durchlaufzeit lautet $z^u = 73$, die untere Schranke lautet $z^\ell = 73 - 3*UB$. Die weiteren Iterationen sind in einer Tabelle zusammengefaßt.

| Iterations-nummer | Restriktionen für das Benderssche Unterprogramm | | | | | | | | | | | Obere Schranke | Neuer Lösungsvektor | Untere Schranke |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | z | $\delta_{121}$ | $\delta_{131}$ | $\delta_{231}$ | $\delta_{122}$ | $\delta_{132}$ | $\delta_{232}$ | $\delta_{123}$ | $\delta_{133}$ | $\delta_{233}$ | $\geqq$ rechte Seite | $z^u$ | $\delta_{I,L,J}$ | $z^\ell$ |
| 0 | - | - | - | - | - | - | - | - | - | - | - | $+\infty$ | $(0,0,0,0,0,0,0,0,0)^T$ | $-\infty$ |
| 1 | 1 | UB | | | | | | | | 2UB | 73 | 73 | $(1,0,0,0,0,0,0,0,1)^T$ | 73-3UB |
| 2 | 1 | | | | UB | | UB | | | | 44 | 44 | $(1,0,0,1,0,1,0,0,1)^T$ | 44-2UB |
| 3 | 0 | | | | -UB | UB | -UB | | | | 13-2UB | 44 | $(1,0,0,1,0,0,0,0,1)^T$ | 44-2UB |
| 4 | 1 | | UB | | -UB | | | | UB | | 43-UB | 43 | $(1,1,0,1,1,1,0,1,1)^T$ | 44-2UB |
| 5 | 1 | | | | -2UB | | -UB | | | | 46-3UB | 43 | $(1,1,0,0,0,1,0,1,1)^T$ | 44-UB |
| 6 | 1 | | | | | 2UB | -2UB | | -UB | | 59-3UB | 43 | $(1,1,0,0,1,1,0,1,1)^T$ | 44-UB |
| 7 | 1 | | | | 2UB | -UB | | | | | 54-UB | 43 | $(1,1,0,1,0,1,0,1,1)^T$ | 46-UB |
| 8 | 1 | | -3UB | | -UB | 3UB | | | -UB | | 54-5UB | 43 | $(1,0,0,1,0,0,0,1,1)^T$ | 46-UB |
| 9 | 1 | | | | -UB | 3UB | | | -UB | | 42-2UB | 42 | $(1,1,0,1,1,0,0,0,1)^T$ | 46-UB |
| 10 | 1 | | | | | -3UB | UB | | UB | | 53-3UB | 42 | $(1,1,0,1,1,0,0,1,1)^T$ | 53-UB |
| 11 | 1 | | -UB | | | -UB | UB | | | | 43-2UB | 42 | $(1,0,0,1,1,0,0,1,1)^T$ | 53-UB |
| 12 | | | 3UB | | | -2UB | UB | | | | 49-2UB | 42 | $(1,0,0,0,0,1,0,0,1)^T$ | 59-UB |
| 13 | 1 | | -3UB | | -UB | UB | | | UB | | 55-4UB | 42 | $(1,1,0,0,0,0,0,0,1)^T$ | 59-UB |
| 14 | 1 | | | | | | -2UB | | UB | | 62-2UB | 42 | - | 42 |

Der bezüglich der Durchlaufzeitminimierung optimale Ablaufplan ist in einem Ganttdiagramm dargestellt.

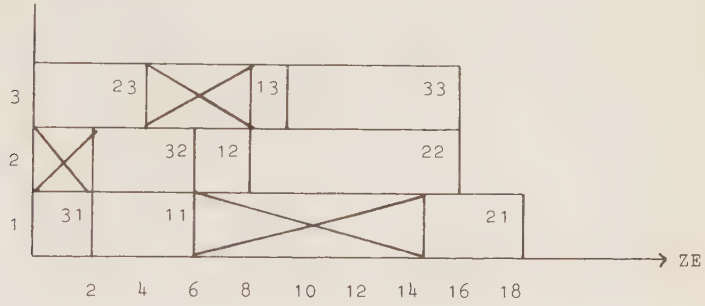



Abbildung 21

Für die Minimierung der Belegungszeit gelten ähnliche Überlegungen, wie für die Minimierung der Durchlaufzeit. Nur werden in dem MANNE-Modell längste Wege von A nach $B_K$ $(K = 1,\ldots,M)$ gesucht. Die für das Benderssche Programm zu konstruierende Restriktion lautet:

$$(4\text{-}28) \quad z \geq \sum_{K=1}^{M} \ell_w(A \cap B_K) - UB^* \sum_{K=1}^{M} \sum_{(IJ,LJ)\in(W(A\cap B_K)\cap D^+)} \text{card}\,(IJ,LJ)$$

$$+ UB^* \sum_{K=1}^{M} \sum_{(IJ,LJ)\in(W(A\cap B_K)\cap D^+)} \delta_{I,L,J}$$

$$- UB^* \sum_{K=1}^{M} \sum_{(IJ,LJ)\in(W(A\cap B_K)\cap D^-)} \delta_{I,L,J}$$

Stellt der Vektor $\delta_{I,L,J}$ keinen zulässigen Ablaufplan dar, so muß im Bendersschen Unterprogramm die Restriktion (4-27) eingeführt werden.

**Beispiel 4.7:**

Für Beispiel 1.2 soll ein optimaler Ablaufplan bezüglich der Zielsetzung „Minimierung der Durchlaufzeit" bestimmt werden.

Setze $z^u = +\infty \qquad z^\ell = -\infty$

$\delta_{I,L,J} = (0,0,0,0,0,0,0,0,0)^T$

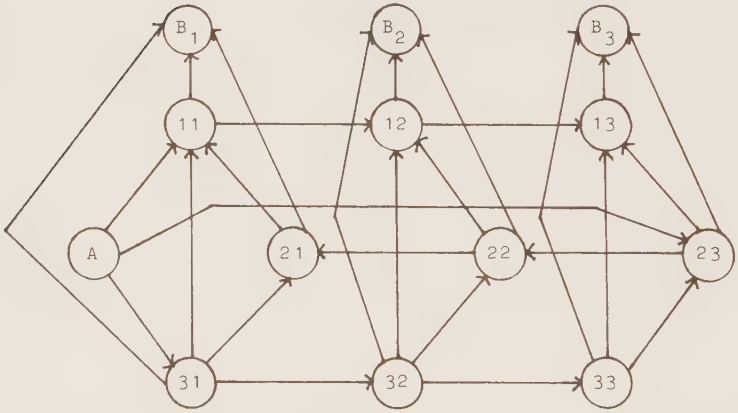



Abbildung 22

Die Bestimmung längster Wege von A nach $B_1$ ergibt:

$<(A,31),(31,32),(32,33),(33,23),(23,22),(22,21),$
$(21,11),(11,B_1)>.$

Die Bestimmung längster Wege von A nach $B_2$ ergibt:

$<(A,31),(31,32),(32,33),(33,23),(23,22),(22,21),$
$(22,21),(21,11),(11,12),(12,B_2)>.$

Die Bestimmung längster Wege von A nach $B_3$ ergibt:

$<(A,31),(31,32),(32,33),(33,23),(23,22),(22,21),$
$(21,11),(11,12),(12,13),(13,B_3)>.$

Aufgrund der theoretischen Herleitung ergibt sich folgendes ganzzahlige Unterprogramm

$$\min z$$
$$z \geq 98 - 3UB*\delta_{121} - 3UB*\delta_{233}.$$

Die Lösung lautet:

$$\delta_{121} = 1, \quad \delta_{233} = 1.$$

Die folgende Tabelle enthält die weiteren Iterationen, die nötig sind, um den optimalen Ablaufplan zu erhalten.

| Iterations-nummer | Restriktionen für das Benderssche Unterprogramm | | | | | | | | | | | | Obere Schranke $z^u$ | Neuer Lösungsvektor $\delta_{I,L,J}$ | Untere Schranke $z^\ell$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | z | $\delta_{121}$ | $\delta_{131}$ | $\delta_{231}$ | $\delta_{122}$ | $\delta_{132}$ | $\delta_{232}$ | $\delta_{123}$ | $\delta_{133}$ | $\delta_{233}$ | $\geqq$ | rechte Seite | | | |
| 0 | - | - | - | - | - | - | - | - | - | - | | - | $+\infty$ | $(0,0,0,0,0,0,0,0,0)^T$ | $-\infty$ |
| 1 | 1 | 3UB | | | | | | | | 3UB | | 98 | 98 | $(1,0,0,0,0,0,0,0,1)^T$ | 98-6UB |
| 2 | 1 | | | | 2UB | | 3UB | | | | | 47 | 47 | $(1,0,0,1,0,1,0,0,1)^T$ | 47-5UB |
| 3 | 0 | | | | -UB | UB | -UB | | | | | 13-2UB | 47 | $(1,0,0,1,1,0,0,0,1)^T$ | 47-5UB |
| 4 | 1 | | | | | -3UB | 2UB | | UB | | | 60-3UB | 47 | $(1,0,0,0,0,1,0,1,1)^T$ | 47-3UB |
| 5 | 1 | | | | | 2UB | -2UB | | -UB | | | 58-3UB | 47 | $(1,0,0,1,0,0,0,0,1)^T$ | 47-2UB |
| 6 | 1 | | | | -2UB | 2UB | | | UB | | | 46-2UB | 46 | $(1,0,0,0,1,1,0,1,1)^T$ | 58-2UB |
| 7 | 1 | | | | 2UB | -2UB | | | | | | 57-2UB | 46 | $(1,0,0,1,1,1,0,1,1)^T$ | 58-2UB |
| 8 | 1 | | 3UB | | -3UB | | +2UB | | | | | 62-5UB | 46 | $(1,1,0,1,1,1,0,1,1)^T$ | 58-2UB |
| 9 | 1 | | | | -3UB | | 2UB | | | | | 56-5UB | 46 | $(1,1,0,1,0,0,0,1,1)^T$ | 58-2UB |
| 10 | 1 | | -3UB | | 2UB | | 3UB | | -UB | | | 65-4UB | 46 | $(1,0,0,1,0,0,0,1,1)^T$ | |
| 11 | 1 | | | | -2UB | 3UB | | | -UB | | | 48-3UB | 46 | $(1,1,0,0,0,1,0,0,1)^T$ | |
| 12 | 1 | | | | | UB | -2UB | | UB | | | 56-2UB | 46 | $(1,0,0,1,1,0,0,1,1)^T$ | |
| 13 | 1 | | 3UB | | | -3UB | 2UB | | | | | 59-3UB | 46 | $(1,1,0,1,1,0,0,1,1)^T$ | |
| 14 | 1 | | | | | -3UB | 2UB | | | | | 53-3UB | 46 | $(1,0,0,1,0,0,0,0,1)^T$ | 46 |

Der bezüglich der Belegungszeitoptimierung optimale Ablaufplan ist in einem Ganttdiagramm dargestellt.

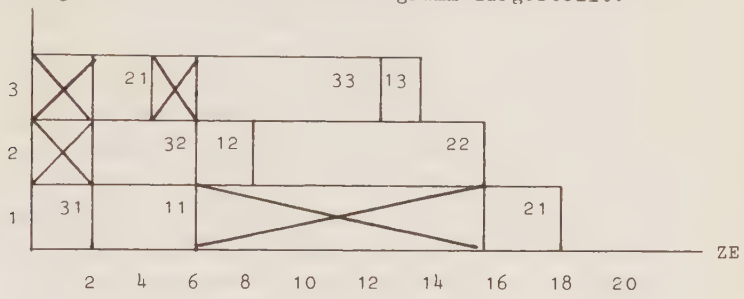



Abbildung 23

In den Beispielen 4.5, 4.6 und 4.7 waren die optimalen Lösungen relativ früh bekannt. Allerdings fehlte der Beweis der Optimalität. Die meisten Iterationsschritte wurden benötigt, um diesen Beweis zu führen. Im sechsten Kapitel werden umfangreiche Testbeispiele mit der Bendersschen Partitionsmethode gelöst.

## 5. Branch und Bound-Methoden für Ablaufplanungsmodelle

Branch und Bound-Methoden spielen eine große Rolle, um diskrete Optimierungsprobleme zu lösen. Ist die Anzahl der zulässigen Lösungen klein, so kann man durch eine Vollenumeration versuchen, die optimale Alternative zu bestimmen. Da diskrete Entscheidungsmodelle zumeist noch kombinatorischen Charakter besitzen, ist die Anzahl der möglichen zulässigen Lösungen meistens sehr groß. Eine Vollenumeration kann also nicht durchgeführt werden, es muß auf eine (implizite) Partialenumeration unter der Bedingung zurückgegriffen werden, daß man die optimale Alternative durch diese Art der Enumeration auch findet. Der wohl erste Branch und Bound-Algorithmus für diskrete Optimierungsprobleme war das Verfahren von LAND und DOIG [37] zur Lösung ganzzahliger linearer Programme. Nach der Veröffentlichung dieses Algorithmus wurde für eine Vielzahl von diskreten Entscheidungsproblemen Branch und Bound-Algorithmen entwickelt (vgl. z.B. MURTY [48, S. 437 ff.], SALKIN [53, S. 165-166].

Bevor die Anwendung von Branch und Bound-Verfahren auf Ablaufplanungsprobleme und ihre Effizienz diskutiert wird, soll das Konzept von Branch und Bound Algorithmen vorgestellt werden.

### 5.1. Das Konzept von Branch und Bound-Algorithmen

Allgemeine Konzepte zur Beschreibung von Branch und Bound-Algorithmen finden sich bei AGIN [1], BALAS [7], BURKARD [13], MITTEN [47] und RINNOOY KAN [51].

Gegeben sei das Optimierungsproblem

(5-1) $\quad \min \quad f(y)$
$\quad\quad\quad\quad$ u.d.N. $y \in Y$,

wobei card $(Y) < \infty$ und $f : Y \to \mathbb{R}$ eine reellwertige Funktion darstellt.

Desweiteren sei eine Obermenge $\bar{Y} \supset Y$ gegeben. Zum Optimierungsproblem (5-1) kann dann das Problem (5-2) konstruiert werden.

(5-2) $\quad \min \quad f(y)$
$\quad\quad\quad\quad$ u.d.N. $y \in \bar{Y}$

Gilt für die optimale Lösung $y^*$ von (5-2), daß $y^* \in Y$ ist, so ist dies auch die optimale Lösung von (5-1).

Durch die Erweiterung der Menge der zulässigen Lösungen Y auf $\bar{Y}$ wird versucht, die Schwierigkeiten, die das diskrete Optimierungsproblem (5-1) besitzt, zu umgehen. Beispielsweise kann man die Ganzzahligkeitsbedingungen für alle Variablen fallenlassen, falls (5-1) ein rein ganzzahliges lineares Programm ist. (5-2) stellt dann ein normales lineares Programm dar, das einfach zu lösen ist.

Ist die optimale Lösung $y^*$ von (5-2) nicht aus Y, so werden mit Hilfe einer Verzweigungsregel (BRANCHING) Teilmengen $\bar{Y}^i$ $(i = 1,\ldots,I_1)$ von $\bar{Y}$ konstruiert, wobei die $\bar{Y}^i$ disjunkte Teilmengen untereinander sind.

Werden die $I_1$ Optimierungsprobleme

(5-3) $\quad \min \quad f(y)$
$\quad\quad\quad\quad$ u.d.N. $y \in \bar{Y}^i$ $\quad\quad$ i fix

gelöst, so erhält man $I_1$ Lösungen $f^i(y^*)$. (5-3) heißt auch Lower-Bounding-Strategie. Gilt für eine Lösung $y^* \in Y$, so wird eine Kontrollgröße $z^u$ (upper bound) gleich diesem Wert gesetzt, anderenfalls wird $z^u = +\infty$ gesetzt. Alle diejenigen Optimierungsprobleme $i \in I_1$ mit $f^i(y^*) \geq z^u$ können aus der weiteren Betrachtung ausgeschlossen werden, da sie höchstens noch einen Zielfunktionswert liefern können, der gleich dem bekannten Wert von $z^u$ ist. Mit der im allgemeinen reduzierten Indexmenge $I_1$ läuft der Optimierungsprozeß also weiter.

Mit Hilfe einer noch zu spezifizierenden Suchstrategie wird die Menge der $I_1$ Optimierungsprobleme weiter untersucht, d.h., es werden zu den Restriktionen von (5-3) weitere Bedingungen hinzugefügt und die neuen Zielwerte berechnet. Es erfolgt wiederum ein Vergleich, ob die neuen Zielwerte größer oder gleich der bekannten oberen Schranke $z^u$ ist. Wiederum wird auch kontrolliert, ob die neuen Werte größer oder gleich der bekannten Fall, so wird $z^u$ gleich $f(y^*)$ gesetzt. Es können alle Teilprobleme von der weiteren Betrachtung ausgeschlossen werden, die Zielwerte größer oder gleich $z^u$ besitzen. Das Verfahren endet genau dann, wenn die Menge der zu untersuchenden Optimierungsprobleme leer ist.

Branch und Bound-Algorithmen unterscheiden sich in drei Bedingungen:
1. der Lower-Bounding-Strategie,
2. der Verzweigungsstrategie und
3. der Suchstrategie.

Bemerkung 5.1:
Bei dem in Kapitel 4 vorgestellten MANNE-Modell bereitet die Menge der disjunktiven Nebenbedingungen die größten numerischen Schwierigkeiten. Eine mögliche Art einer

„Lower Bounding"-Strategie besteht darin, diese Restriktionen teilweise oder ganz wegzulassen. Man erhält auf diese Weise das Optimierungsproblem (5-2), welches im allgemeinen einfacher zu lösen ist. Dieses Verfahren, Restriktionen wegzulassen, ist in der Literatur als Relaxing-Methode bekannt. Nach Konstruktion der Mengen $\bar{Y}^i$ ($i \in I_1$) sind diese paarweise disjunkt, ihre Vereinigung bildet im ersten Verzweigungsschritt gerade $\bar{Y}$. Die Mengen $\bar{Y}^i$ heißen auch „Kandidaten"-Probleme. Speziell auf Ablaufplanungsprobleme bezogen entstehen solche „Kandidaten" (Knoten) durch Hinzufügen von disjunktiven Nebenbedingungen. Jede Lösung dieser Probleme erbringt eine untere Schranke für den Zielfunktionswert, den man mit dem gerade behandelten Teilproblem erreichen kann. Es kann geschehen, daß diese untere Schranke größer oder gleich einer bekannten Lösung von (5-1) ist. Diese Teilprobleme können von der weiteren Untersuchung ausgeschlossen werden. Die Mengen $\bar{Y}^i$ ($i \in I_1$) werden nun weiter zerlegt, indem neue disjunktive Nebenbedingungen bei Ablaufplanungsproblemen hinzugefügt werden. Diese Vorgehensweise heißt auch BRANCHING.

Die Suchstrategien geben an, in welcher Reihenfolge die Kandidaten-Probleme behandelt werden sollen. Zwei mögliche Suchstrategien werden näher erläutert (vgl. AHO/HOPCROFT/ULLMANN [2], NOLTEMEIER [50], RINNOOY KAN [51]).

a) (JUMPTRACK)
   Es wird immer das Kandidaten-Problem untersucht, das die kleinste untere Schranke besitzt. Auf diese Weise gelangt man relativ schnell zu einer zulässigen Lösung für das Optimierungsproblem (5-1).

b) (BACKTRACK)
   Es werden jeweils auf einer Stufe alle Möglichkeiten überprüft, bevor dann weitere Kandidaten untersucht werden. Ist eine obere Schranke für den optimalen

Zielwert bekannt, so können relativ frühzeitig ganze Lösungszweige von der weiteren Betrachtung ausgeschlossen werden.

Es kann für Branch und Bound-Methoden keine allgemein gültige Verzweigungsregel angegeben werden, so daß die Anzahl der erzeugten Teilprobleme minimal wird. FOX/LENSTRA/RINNOOY KAN/SCHRAGE geben in [22] Beispiele dafür an, daß die JUMPTRACK-Strategie, die in der Literatur über Branch und Bound-Algorithmen als optimal dargestellt wurde, nicht unbedingt die beste Verzweigungsregel zu sein braucht.

## 5.2. Der Branch und Bound-Algorithmus von GREENBERG

1968 gab GREENBERG in [26] einen Branch und Bound-Algorithmus an, der das MANNE-Modell wesentlich benutzt.

Hier sei nochmals das MANNE-Modell für die Zielsetzung „Minimierung der Zykluszeit" angeführt.

$$
\begin{aligned}
(5\text{-}4)\quad & \min \quad Z \\
& \text{u.d.N.} \quad h_{I[J]} + p_{I[J]} \leqq h_{I[J+1]} \qquad (I = 1,\ldots,N;\ J = 1,\ldots,M-1) \\
& \qquad h_{IJ} + p_{IJ} \leqq h_{I+L,J} + UB*(1-\delta_{I,L,J}) \\
& \qquad h_{I+L,J} + p_{I+L,J} \leqq h_{IJ} + UB*\delta_{I,L,J} \\
& \qquad (J = 1,\ldots,M;\ I = 1,\ldots,N-1;\ L = 1,\ldots,N-I) \\
& \qquad h_{I[M]} + p_{I[M]} \leqq z \qquad (I = 1,\ldots,N) \\
& \qquad z,\ h_{IJ} \geqq 0 \qquad (I = 1,\ldots,N;\ J = 1,\ldots,M) \\
& \qquad \delta_{I,L,J} \in \{0,1\} \qquad (J = 1,\ldots,M;\ I = 1,\ldots,N-1;\ L = 1,\ldots,N-I)
\end{aligned}
$$

Läßt man die disjunktiven Restriktionen von (5-4) weg, so reduziert sich (5-4) auf ein gewöhnliches lineares Programm mit Netzwerkstruktur. Der Zulässigkeitsbereich dieses Problems ist eine Obermenge der zulässigen Lösungen von (5-4).

Die Lösung des linearen Programms stellt im allgemeinen keinen zulässigen Ablaufplan dar, da eine Maschine mit mehreren Aufträgen gleichzeitig belegt sein kann. An dieser Stelle wird dann dem linearen Programm jeweils die erste der disjunktiven Restriktionen von (5-4) hinzugefügt, indem $\delta_{I,L,J}$ gleich Eins gesetzt wird. Dann wird dem ursprünglich linearen Programm die zweite der disjunktiven Restriktionen hinzugefügt, indem $\delta_{I,L,J}$ gleich Null gesetzt wird.

Während mit dem Bendersschen Verfahren in jedem Iterationsschritt ein neuer zulässiger Ablaufplan erzeugt wird, falls kein Zyklus vorliegt, fügt GREENBERG dem bisher nicht zulässigen Ablaufplan eine weitere Restriktion hinzu, um so einen zulässigen Plan zu konstruieren.

Im folgenden Algorithmus 10 von GREENBERG wird *allgemein*, d.h. ohne Verzweigungsregel und ohne die Zielfunktionen, das Branch und Bound-Verfahren beschrieben. Die Ausführungen in Kapitel 4 zeigen, wie die linearen Programme als längste Wegeprobleme interpretiert werden können.

Algorithmus 10:

*Name* Branch und Bound-Verfahren von GREENBERG.

*Voraussetzung:* Gegeben sei der Graph
OP := (V,C,P).

*Initialisierung:* LISTE := VERWEISLISTE := $D^* = \{\emptyset\}$.
Setze $z^u := +\infty$.

*Iteration:*
1. Löse das Ablaufplanungsproblem OP.
2. Speichere in LISTE alle Konflikte zwischen den Aufträgen I und K ($I,K \in \{1,\ldots,N\}, I \neq K$) auf Maschine J ($\forall J \in \{1,\ldots,M\}$). Setze für alle Elemente (I,K,J). VERWEISLISTE (I,K,J) = 0.

*STOPKRITERIUM*
3. Ist LISTE = $\{\emptyset\}$? Wenn ja, STOP.

*Verzweigungsregel*
Wenn nein, wähle ein Element (I,K,J) $\in$ LISTE.
4. $D^* := D^* \cup$ VERWEISLISTE (I,K,J).
5. Bilde die beiden Optimierungsprobleme
OP1 := $\{V, C \in D^* \cup (IJ,KJ), P\}$,
OP2 := $\{V, C \in D^* \cup (KJ,IJ,P)\}$.
6. Überprüfe mit Algorithmus 1 die Optimierungsprobleme OP1 und OP2 auf Zyklenfreiheit. Falls ein Zyklus im Graphen (Netzwerk) ist, so braucht das entsprechende Problem nicht mehr betrachtet zu werden. Markiere dies in LISTE.
7. Löse die Optimierungsprobleme. Die resultierenden Zielwerte seien mit ZF1 bzw. ZF2 bezeichnet.

*Prüfung auf Zulässigkeit*
8. Ermittle alle Konflikte und speichere sie in LISTE, speichere in VERWEISLISTE die festgelegten Reihenfolgen.
Sind bei einem Problem keine Konflikte mehr, so korrigiere unter Umständen $z^u$, falls ein Ablaufplan erzeugt wurde, der eine kleinere obere Schranke liefert als bisher bekannt war.
9. Falls $z^u \leqq$ ZF1 oder $z^u \leqq$ ZF2, so braucht OP1 bzw. OP2 nicht weiter verfolgt zu werden. Streiche in LISTE die entsprechenden Probleme.

10. Markiere in LISTE die Probleme, die in Schritt 5 behandelt werden, als abgearbeitet.
11. Gehe zu Schritt 3.

Bevor der Rechenaufwand dieses Algorithmus diskutiert werden soll, wird die Vorgehensweise demonstriert.

Beispiel 5.1:

Gegeben sei das Beispiel 1.2 mit den Matrizen

$$P = \begin{pmatrix} 4 & 2 & 1 \\ 3 & 7 & 5 \\ 2 & 4 & 6 \end{pmatrix}, \quad Q = \begin{pmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \\ 1 & 2 & 3 \end{pmatrix}.$$

Es soll die optimale Zykluszeit ermittelt werden, wobei als Verzweigungsstrategie eine JUMPTRACK-Strategie verwendet wird.

*Initialisierung:* LISTE := VERWEISLISTE := {∅},
$z^u := +\infty$.

Schritt 0: Zielwert = 15 ZE.
Schritt 1: Es ergeben sich 5 Konflikte.

| Nr. | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| LISTE | (1,3,1) | (1,2,2) | (1,3,2) | (2,3,2) | (1,3,3) |
| VERWEISLISTE | 0 | 0 | 0 | 0 | 0 |

Schritt 2: Wähle (1,3,1).
Schritt 3: $D^* := D^*$.
Schritt 4: OP1 := OP ∪ (1,3,1),
OP2 := OP ∪ (3,1,1).
Schritt 5: Kein Zyklus in den beiden Netzwerken.
Schritt 6: ZF1 = 16,
ZF2 = 15.

Bedingung 8 ist nicht erfüllt.
Schritt 9: Die bei OP1 und OP2 vorhandenen Konflikte werden gespeichert.

| Nr. | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|
| LISTE | (1,2,2) | (2,3,2) | (1,2,2) | (2,3,2) | (1,3,3) |
| VERWEISLISTE | (1,3,1) | (1,3,1) | (3,1,1) | (3,1,1) | (3,1,1) |

Problem 2 wird weiter untersucht.
Schritt 4: OP1 := OP U (1,2,2),
OP2 := OP U (2,1,2).
Schritt 6: ZF1 = 16,
ZF2 = 15.
Schritt 9:

| Nr. | 11 | 12 | 13 | 14 |
|---|---|---|---|---|
| LISTE | (1,3,1) | (1,3,3) | (1,3,1) | (2,3,2) |
| VERWEISLISTE | (1,2,2) | (1,2,2) | (2,1,2) | (2,1,2) |

Die Ergebnisse für die Probleme 3, 4 und 5 werden gekürzt dargestellt.

Schritt 6 für Knoten 3: ZF1 = 16, ZF2 = 15,
für Knoten 4: ZF1 = 22, ZF2 = 16,
für Knoten 5: ZF1 = 15, ZF2 = 15.

| Nr. | 15 | 16 | 17 | 18 | 19 |
|---|---|---|---|---|---|
| LISTE | (1,3,2) | (1,2,2) | (2,3,2) | (1,3,1) | (1,2,2) |
| VERWEISLISTE | (1,3,2) | (1,3,2) | (1,3,2) | (3,1,2) | (3,1,2) |
| | 20 | 21 | 22 | 23 | 24 |
| | (2,3,2) | (1,3,3) | (1,3,1) | (1,2,2) | (1,3,1) |
| | (3,1,2) | (3,1,2) | (2,3,2) | (2,3,2) | (3,2,2) |
| | 25 | 26 | 27 | 28 | 29 |
| | (1,3,2) | (1,3,3) | (1,3,1) | (1,2,2) | (1,3,3) |
| | (3,2,2) | (3,2,2) | (1,3,3) | (1,3,3) | (1,3,3) |

| | 30 | 31 | 32 | 33 | 34 |
|---|---|---|---|---|---|
| | (2,3,2) | (1,3,1) | (1,2,2) | (1,3,2) | (2,3,2) |
| | (1,3,3) | (3,1,3) | (3,1,3) | (3,1,3) | (3,1,3) |

Damit ist die erste Stufe vollkommen abgearbeitet (Probleme 1 bis 5). Da eine JUMPTRACK-Strategie verwendet werden soll, sind alle Probleme von Interesse, die bisher einen Zielfunktionswert von 15 ZE aufweisen, d.h. die Aufgaben 8,9,10,13,14,18,19,20,21,27,28,29,30,31,32,33,34.

Folgende Zielwerte ergeben sich für diese Probleme:

| Nr. des Knotens | ZF1 | ZF2 |
|---|---|---|
| 8 | 18 | 15 |
| 9 | 20 | 16 |
| 10 | 15 | 15 |
| 13 | 16 | 15 |
| 14 | 22 | 16 |
| 18 | 16 | 15 |
| 19 | 18 | 15 |
| 20 | 22 | 16 |
| 21 | 15 | 15 |
| 27 | 16 | 15 |
| 28 | 16 | 21 |
| 29 | 16 | 15 |
| 30 | 22 | 16 |
| 31 | 17 | 15 |
| 32 | 16 | 15 |
| 33 | 15 | 15 |
| 34 | 22 | 16 |

Werden die zum Zielwert Z = 15 gehörenden neuen Probleme gespeichert, so ergeben sich insgesamt 30 neue Optimierungsaufgaben.

| Nr. | 35 | 36 | 37 | 38 | 39 |
|---|---|---|---|---|---|
| LISTE<br>VERWEISLISTE | (2,3,2)<br>(3,1,1)<br>(2,1,2) | (1,2,2)<br>(3,1,1)<br>(1,3,3) | (1,2,2)<br>(3,1,1)<br>(3,1,3) | (2,3,2)<br>(3,1,1)<br>(2,1,2) | (1,2,2)<br>(3,1,1)<br>(2,1,2) |
| | 40 | 41 | 42 | 43 | 44 |
| | (2,3,2)<br>(3,1,1)<br>(3,1,2) | (1,3,3)<br>(3,1,1)<br>(3,1,2) | (1,3,1)<br>(2,1,2)<br>(3,1,2) | (2,3,2)<br>(2,1,2)<br>(3,1,2) | (1,3,1)<br>(3,1,2)<br>(1,3,3) |
| | 45 | 46 | 47 | 48 | 49 |
| | (1,2,2)<br>(3,1,2)<br>(1,3,3) | (2,3,2)<br>(3,1,2)<br>(1,3,3) | (1,3,1)<br>(3,1,2)<br>(3,1,3) | (1,2,2)<br>(3,1,2)<br>(3,1,3) | (2,3,2)<br>(3,1,2)<br>(3,1,3) |
| | 50 | 51 | 52 | 53 | 54 |
| | (1,2,2)<br>(3,1,1)<br>(1,3,3) | (2,3,2)<br>(3,1,1)<br>(1,3,3) | (1,3,1)<br>(3,1,2)<br>(1,3,3) | (1,2,2)<br>(3,1,2)<br>(1,3,3) | (2,3,2)<br>(3,1,2)<br>(1,3,3) |
| | 55 | 56 | 57 | 58 | 59 |
| | (1,2,2)<br>(3,1,1)<br>(3,1,3) | (2,3,2)<br>(3,1,1)<br>(3,1,3) | (1,3,1)<br>(2,1,2)<br>(3,1,3) | (2,3,2)<br>(2,1,2)<br>(3,1,3) | (1,3,1)<br>(1,3,2)<br>(3,1,3) |
| | 60 | 61 | 62 | 63 | 64 |
| | (1,2,2)<br>(1,3,2)<br>(3,1,3) | (2,3,2)<br>(1,3,2)<br>(3,1,3) | (1,3,1)<br>(3,1,2)<br>(3,1,3) | (1,2,2)<br>(3,1,2)<br>(3,1,3) | (2,3,2)<br>(3,1,2)<br>(3,1,3) |

Vergleicht man die festgelegten Reihenfolgen in VERWEIS-LISTE und die disjunktiven Bedingungen in LISTE, so kann festgestellt werden, daß einige Knoten redundant sind, weil sie mehrfach generiert werden. Dies gilt z.B. für die Knoten 35 und 38. Desweiteren gilt dies für die Knoten 36 und 50, 37 und 55, 44 und 52, 45 und 53, 46 und 54, 47 und 62, 48 und 63 und 49 und 64. Von den generierten Knoten hätten also 9 Knoten, d.h. 18 Probleme, nicht untersucht werden müssen, wenn man im GREENBERG-Algorithmus eine zusätzliche Redundanzprüfung vorgenommen hätte.

Folgende Zielwerte ergeben sich für diese Probleme:

| Nr. des Knotens | ZF1 | ZF2 | |
|---|---|---|---|
| 35 | 26 | 16 | (zulässiger Ablaufplan) |
| 36 | 18 | 21 | |
| 37 | 18 | 15 | |
| 38 | 26 | 16 | |
| 39 | 18 | 15 | |
| 40 | 22 | 16 | |
| 41 | 15 | 15 | |
| 42 | 16 | 15 | |
| 43 | 22 | 16 | |
| 44 | 19 | 15 | |
| 45 | 18 | 21 | |
| 46 | 25 | 16 | |
| 47 | 17 | 15 | |
| 48 | 18 | 15 | |
| 49 | 23 | 16 | |
| 50 | 18 | 21 | |
| 51 | 22 | 16 | |
| 52 | 19 | 15 | |
| 53 | 18 | 21 | |
| 54 | 25 | 16 | |
| 55 | 18 | 15 | |
| 56 | 23 | 16 | |
| 57 | 17 | 15 | |
| 58 | 23 | 16 | |
| 59 | 17 | 19 | |
| 60 | 17 | 25 | |
| 61 | 23 | 20 | |
| 62 | 17 | 15 | |
| 63 | 18 | 15 | |
| 64 | 23 | 16 | |

Es können alle Probleme, die einen größeren Zielwert als 16 ZE besitzen, gestrichen werden. Es werden nur noch die Probleme mit einem Zielwert von 15 ZE betrachtet.

| Nr. | 65 | 66 | 67 | 68 | 69 |
|---|---|---|---|---|---|
| LISTE | (2,3,2) | (2,3,2) | (1,2,2) | (2,3,2) | (1,2,2) |
| VERWEISLISTE | (3,1,1)<br>(2,1,2)<br>(3,1,3) | (3,1,1)<br>(2,1,2)<br>(3,1,2) | (3,1,1)<br>(3,1,2)<br>(1,3,3) | (3,1,1)<br>(3,1,2)<br>(1,3,3) | (3,1,1)<br>(3,1,2)<br>(3,1,3) |
| | 70 | 71 | 72 | 73 | 74 |
| | (2,3,2) | (2,3,2) | (1,2,2) | (1,2,2) | (1,3,1) |
| | (3,1,1)<br>(3,1,2)<br>(3,1,3) | (3,1,1)<br>(2,1,2)<br>(3,1,2) | (3,1,1)<br>(3,1,2)<br>(1,3,3) | (3,1,1)<br>(3,1,2)<br>(3,1,3) | (2,1,2)<br>(3,1,2)<br>(3,1,3) |
| | 75 | 76 | 77 | 78 | 79 |
| | (2,3,2) | (1,2,2) | (2,3,2) | (2,3,2) | (1,2,2) |
| | (2,1,2)<br>(3,1,2)<br>(3,1,3) | (3,1,1)<br>(3,1,2)<br>(1,3,3) | (3,1,1)<br>(3,1,2)<br>(1,3,2) | (3,1,1)<br>(2,1,2)<br>(3,1,3) | (3,1,1)<br>(3,1,2)<br>(3,1,3) |
| | 80 | 81 | 82 | | |
| | (2,3,2) | (1,3,1) | (2,3,2) | | |
| | (3,1,1)<br>(3,1,2)<br>(3,1,3) | (2,1,2)<br>(3,1,2)<br>(3,1,3) | (2,1,2)<br>(3,2,1)<br>(3,1,3) | | |

Identisch sind wieder die Aufgaben 65 und 78, 66 und 71, 67 und 76, 69, 73 und 79, 70 und 80 und 74 und 81.

Folgende Zielwerte ergeben sich für diese Probleme:

| Nr. des Knotens | ZF1 | ZF2 |
|---|---|---|
| 65 | 23 | 16 |
| 66 | 22 | 16 |
| 67 | 18 | 21 |
| 68 | 25 | 16 |
| 69 | 18 | 15 |
| 70 | 23 | 16 |
| 71 | 22 | 16 |
| 72 | 18 | 21 |
| 73 | 18 | 15 |
| 74 | 17 | 15 |
| 75 | 23 | 16 |
| 76 | 18 | 21 |
| 77 | 23 | 16 |
| 78 | 23 | 16 |
| 79 | 18 | 15 |
| 80 | 23 | 16 |
| 81 | 17 | 15 |
| 82 | 23 | 16 |

Alle Aufgaben mit dem Zielwert Z = 15 ZE müssen noch untersucht werden, da sie noch nicht zulässige Ablaufpläne darstellen.

| Nr. | 83 | 84 | 85 | 86 | 87 |
|---|---|---|---|---|---|
| LISTE | (2,3,2) | (2,3,2) | (2,3,2) | (2,3,2) | (2,3,2) |
| VERWEISLISTE | (3,1,1)<br>(2,1,2)<br>(3,1,2)<br>(3,1,3) | (3,1,1)<br>(2,1,2)<br>(3,1,2)<br>(3,1,3) | (3,1,1)<br>(2,1,2)<br>(3,1,2)<br>(1,3,3) | (3,1,1)<br>(2,1,2)<br>(3,1,2)<br>(3,1,3) | (3,1,1)<br>(2,1,2)<br>(3,1,2)<br>(3,1,3) |

Es sind lediglich noch zwei Optimierungsaufgaben unterschiedlich.

Für das Problem 83 (und damit für die Aufgaben 84, 86 und 87) ergeben sich die Zielwerte:

ZF1 = 23 ZE und ZF2 = 16 ZE.

Für die Aufgabe 84 ergeben sich die Zielwerte:

ZF1 = 25 ZE und ZF2 = 22 ZE.

Damit stoppt das Verfahren von GREENBERG. Eine von mehreren Lösungen ist der Ablaufplan, der mit Problem 83 erzeugt wird. Der optimale Plan ist in einem Ganttdiagramm in Abbildung 24 dargestellt.

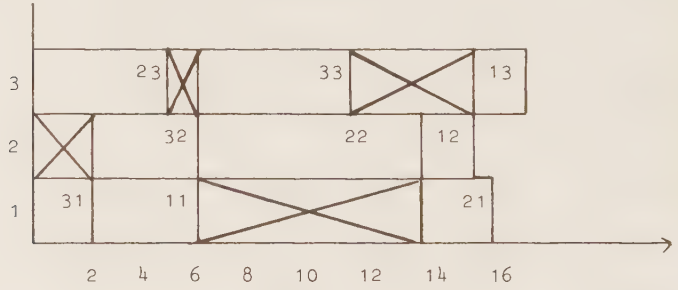



Abbildung 24

Wie an diesem relativ kleinen Beispiel klar wird, ist der Rechenaufwand sehr hoch. Das Verhalten des Algorithmus von GREENBERG im schlechtesten Fall ist in Satz 5.1 dargestellt.

Satz 5.1:
Die Anzahl der maximal zu lösenden Optimierungsaufgaben beträgt $N*(N-1)*M*(2^{N*(N-1)*M/2+1}-1)$.

Beweis: Da N Aufträge auf M Maschinen zu bearbeiten sind, wird eine vollständige Reihenfolge aller Operationen durch $N*(N-1)*M/2$ Binärvariablen beschrieben. Daraus ergibt sich also als maximale „Tiefe" des Branch und Bound-Verfahrens von $N*(N-1)*M/2$. Wird das Optimierungsproblem zu Anfang ohne Auftragsreihenfolgebedingungen für die M Maschinen gelöst, so können maximal $N*(N-1)*M/2$ Konflikte auf den einzelnen Maschinen entstehen, d.h., es müssen $N*(N-1)*M$ Optimierungsaufgaben gelöst werden. In der nächsten Iterationsstufe beim Algorithmus von GREENBERG müssen dann wieder maximal $2*N*(N-1)*M$ Aufgaben betrachtet werden. Insgesamt ergibt sich also für die maximal zu lösenden Optimierungsaufgaben:

$$N*(N-1)*M*\sum_{i=0}^{N*(N-1)*M/2} 2^i = N*(N-1)*M*(2^{N*(N-1)*M/2+1}-1)$$

Bemerkung 5.1:
Satz 5.1 beschreibt das Verhalten des Algorithmus von GREENBERG im schlechtesten Fall, was gleichbedeutend mit einer Vollenumeration ist. Sehr viele Optimierungsprobleme sind als redundant anzusehen, da bestimmte Reihenfolgen mehrfach generiert werden. In Satz 5.1 wird außerdem die Tatsache außer acht gelassen, daß ganze Lösungswege vernachlässigt werden können, sobald eine erste zulässige Lösung gefunden wurde. Dennoch existiert bis heute keine eingängige allgemeine Theorie darüber, wie sich die Laufzeit von Branch und Bound-Algorithmen bestimmen läßt. Diese Probleme, die man übrigens auch mit dem Simplex-Algorithmus hinsichtlich ihrer Laufzeit hat, lassen

sich mit den heutigen mathematischen Hilfsmitteln nur über umfangreiche statistische Computerexperimente in etwa erfassen. Es muß allerdings wiederum betont werden, daß aus diesen Erfahrungen nicht unbedingt allgemeine Schlüsse ableitbar sind.

Nach Satz 5.1 hätten für Beispiel 5.1 maximal 18.414 Optimierungsprobleme gelöst werden müssen. Es brauchten aber „nur" 87 Probleme betrachtet zu werden. Dieses Beispiel zeigt, daß die Bestimmung des maximal möglichen Rechenaufwandes für ein Branch und Bound-Verfahren nicht geeignet ist, um den Rechenverlauf zu beschreiben.

In Algorithmus 10 werden in Schritt 3 Elemente (I,J,K) $\in$ LISTE mit Hilfe einer Verzweigungsregel ausgewählt. Es kann durch die Auswahl passieren, daß Elemente aus LISTE ausgewählt werden, die zu Zyklen in den Ablaufplänen führen. LAGEWEG/LENSTRA/RINNOOY KAN geben in [38, S. 446, Theorem 4] notwendige und hinreichende Bedingungen dafür an, um unzulässige Ablaufpläne zu vermeiden. Dieses Theorem bedeutet für den Rechenverlauf des Algorithmus von GREENBERG eine wirkungsvolle Reduktion des Rechenaufwandes.

Im vorigen Algorithmus wurde die Reihenfolgebeziehung zwischen zwei Operationen auf einer Maschine festgelegt. Nun können aber auch die Positionen in der Auftragsfolge einer Maschine betrachtet werden, d.h., es wird die Position eines Auftrages gegenüber den übrigen N-1 Aufträgen eindeutig festgelegt. Im folgenden Algorithmus wird ein Grundmodell eines Branch und Bound-Verfahrens angegeben, in dem die beschriebene Vorgehensweise realisiert wird.

Algorithmus 11:

*Name:* Grundmodell eines Branch und Bound-Algorithmus.

*Voraussetzung:* Gegeben sei der disjunktive Graph $G = \{V,C,D,P\}$.

*Initialisierung:* LISTE := VERWEISLISTE := HILFSLISTE := AKTUELL := $\{\emptyset\}$. Setze $z^u := +\infty$.

*Iteration:*

1. Fasse die ersten Bearbeitungen der N Aufträge in LISTE zusammen.

*STOPKRITERIUM:*

2. Ist LISTE = $\{\emptyset\}$? Wenn ja, STOP.
3. Plane jede Operation IJ aus LISTE in der Auftragsfolge der Maschine J ein unter Beachtung der schon festgelegten Reihenfolge, die in AKTUELL gespeichert ist.
4. Falls der resultierende Zielwert für das in Schritt 3 behandelte Problem größer oder gleich $z^u$ ist, so streiche in LISTE die Operation IJ.
5. Ist der erzeugte Ablaufplan zulässig und sein Zielwert echt kleiner als $z^u$, so korrigiere $z^u$ und entferne aus LISTE diejenigen Elemente, die zu einem größeren Wert führen als $z^u$.
6. Speichere die festgelegte Reihenfolge des Auftrags I auf Maschine J in VERWEISLISTE.
7. In HILFSLISTE werden die Nachfolgeoperation von IJ und alle Operationen aus LISTE gespeichert, die von IJ verschieden sind.

*Verzweigungsregel:*

8. Wähle mit einer Verzweigungsstrategie das nächste zu untersuchende Problem aus.
9. Speichere alle Operationen aus HILFSLISTE in LISTE und die festgelegten Reihenfolgen, die durch VERWEISLISTE beschrieben werden, in AKTUELL.
10. Gehe zu Schritt 2.

An Beispiel 5.1 wird der Algorithmus 11 mit einer JUMPTRACK-Strategie demonstriert.

LISTE := {11,23,11}.

Für die drei Optimierungsprobleme ergeben sich die Zielwerte

$Z_1$ = 16 ZE, $Z_2$ = 15 ZE, $Z_3$ = 15 ZE.

Für die Probleme 1, 2 und 3 ergibt sich HILFSLISTE:

| Nr. | |
|---|---|
| 1 | {23,31,12} |
| 2 | {22,11,31} |
| 3 | {11,23,32} |

Da eine JUMPTRACK-Strategie verwendet wird, betrachtet man zunächst Problem 2 und anschließend Problem 3.

AKTUELL := Auftrag 2 steht an erster Stelle der Auftragsfolge von Maschine 3.
:= {(2,3,1)}
LISTE := {22,11,31}.

Es ergeben sich die Werte

$Z_4$ = 22 ZE, $Z_5$ = 16 ZE, $Z_6$ = 15 ZE.

AKTUELL := {(3,1,1)}.
LISTE := {11,23,32}.

$Z_7$ = 15 ZE, $Z_8$ = 15 ZE, $Z_9$ = 16 ZE.

Die Probleme 6, 7 und 8 haben einen Zielwert von 15 ZE.

HILFSLISTE sieht folgendermaßen aus:

| Nr. | |
|---|---|
| 6 | {22,11,32} |
| 7 | {12,23,32} |
| 8 | {11,22,32} |

Für Problem 6 hat AKTUELL die Form:

$$\{(2,3,1),\ (3,1,1)\}$$

$Z_{10}$ = 20 ZE, $Z_{11}$ = 15 ZE, $Z_{12}$ = 16 ZE.

AKTUELL enthält für Problem 7 die Elemente

$$\{(3,1,1),\ (1,1,2)\}$$

$Z_{13}$ = 18 ZE, $Z_{14}$ = 15 ZE, $Z_{15}$ = 16 ZE.

Bezüglich Problem 8 hat AKTUELL das Aussehen

$$\{(3,1,1),\ (2,3,1)\}.$$

Da die Probleme 6 und 8 die gleichen Reihenfolgen enthalten, braucht Problem 8 nicht weiter untersucht zu werden.

Es werden jetzt die Optimierungsaufgaben 11 und 14 weiter untersucht. HILFSLISTE hat die Form:

| Nr. | |
|---|---|
| 11 | {(22,12,32)} |
| 14 | {(12,22,32)} |

AKTUELL hat für Problem 11 die Form

{(2,3,1), (3,1,1), (1,1,2)}

$Z_{16}$ = 22 ZE, $Z_{17}$ = 18 ZE, $Z_{18}$ = 16 ZE.

Da AKTUELL und HILFSLISTE für Problem 14 gleich sind, werden diese Probleme nicht untersucht.

Es wird lediglich Optimierungsaufgabe 18 betrachtet.

AKTUELL := {(3,1,1), (1,1,2), (2,3,1), (3,2,1)}.
HILFSLISTE := {(12,22,33)}.

$Z_{19}$ = 18 ZE, $Z_{20}$ = 16 ZE, $Z_{21}$ = 16 ZE.

Das in Problem 20 behandelte Optimierungsproblem liefert einen zulässigen Ablaufplan. Da der Zielwert von 16 ZE kleiner oder gleich allen anderen Werten ist, bricht das Verfahren ab.

Satz 5.2 beschreibt den Rechenaufwand für Algorithmus 11.

Satz 5.2:
Die Anzahl der maximal zu lösenden Optimierungsprobleme beträgt:

$$\frac{1-N^{N*M-N+1}}{1-N} + N^{N*M-N} * \frac{(N-1)*N}{2}.$$

Beweis: Da N*M Operationen bearbeitet werden müssen, ergibt sich als die maximale „Tiefe" des Verfahrens N*M. In der ersten Iterationsstufe müssen N Aufträge eingeplant werden, in der zweiten Stufe wiederum N Aufträge, d.h., es sind jetzt $N^2$ Optimierungsaufgaben zu lösen. In der (N*M-N)-ten Iterationsstufe müssen maximal $N^{N*M-N}$ Aufgaben betrachtet werden. Da dann eine Operation

keine technologische Nachfolgeoperation mehr besitzt, müssen also nur noch $N^{N*M-N}*(N-1)$ Probleme untersucht werden. Wird diese Überlegung bis zur (N*M)-ten Stufe weitergeführt, so ergibt sich also für die Anzahl der maximal zu lösenden Optimierungsaufgaben

$$\sum_{i=1}^{N*M-N} N^i + N^{N*M-N} * \sum_{i=0}^{N-1} i$$

$$= \frac{1-N^{N*M-N+1}}{1-N} + N^{N*M-N} * \frac{(N-1)*N}{2}.$$

Bemerkung 5.2:

Satz 5.2 beschreibt das Verhalten des Algorithmus 11 bei einer Vollenumeration, das der schlechteste Fall für dieses Verfahren ist. Setzt man N und M gleich 3, so müßten maximal 3.280 Optimierungsaufgaben gelöst werden.

Der Rechenverlauf, der in Algorithmus 11 beschrieben wird, ist identisch mit der Konstruktion von semiaktiven Ablaufplänen. In Kapitel 3 wurde ein Verfahren angegeben, mit dem ein beliebiger, semiaktiver Plan erzeugt wurde, während hier das Ziel gesetzt ist, die optimale Lösung zu finden. Die Anzahl der zu untersuchenden Optimierungsprobleme kann unter Umständen im beschriebenen Algorithmus auch noch dadurch reduziert werden, wenn anstatt des Konzeptes semiaktive Ablaufpläne zu erzeugen, aktive Pläne konstruiert werden. In LAGEWEG/LENSTRA/RINNOOY KAN [38] ist diese Vorgehensweise mit anderen Strategien numerisch untersucht worden.

## 6. Numerische Untersuchungen

### 6.1. Numerische Ergebnisse mit der Benderssschen Partitionsmethode

Um die Eigenschaften der Benderssschen Partitionsmethode zur Lösung von Ablaufplanungsproblemen zu studieren, wurden 10 Testbeispiele mit 4 Aufträgen und 3 Maschinen untersucht.

Bei den ersten fünf Testbeispielen ist die Reihenfolgematrix

$$Q = \begin{pmatrix} 1 & 3 & 2 \\ 1 & 2 & 3 \\ 3 & 1 & 2 \\ 2 & 3 & 1 \end{pmatrix}$$

konstant. Für die Bearbeitungszeitenmatrix P werden die folgenden Werte gewählt:

1. Testbeispiel:
$$P = \begin{pmatrix} 4 & 5 & 7 \\ 6 & 3 & 3 \\ 2 & 4 & 6 \\ 7 & 5 & 8 \end{pmatrix}$$

2. Testbeispiel:
$$P = \begin{pmatrix} 3 & 7 & 8 \\ 4 & 5 & 6 \\ 2 & 1 & 9 \\ 10 & 10 & 5 \end{pmatrix}$$

3. Testbeispiel:
$$P = \begin{pmatrix} 7 & 4 & 6 \\ 3 & 6 & 9 \\ 5 & 10 & 6 \\ 7 & 4 & 3 \end{pmatrix}$$

4. Testbeispiel:
$$P = \begin{pmatrix} 4 & 4 & 7 \\ 3 & 1 & 4 \\ 2 & 5 & 9 \\ 3 & 7 & 8 \end{pmatrix}$$

5. Testbeispiel:

$$P = \begin{pmatrix} 8 & 5 & 3 \\ 7 & 6 & 9 \\ 4 & 6 & 8 \\ 5 & 7 & 9 \end{pmatrix}$$

Bei den übrigen Testbeispielen wird jedoch die Bearbeitungszeitenmatrix konstant gesetzt.

$$P = \begin{pmatrix} 3 & 4 & 9 \\ 5 & 3 & 2 \\ 8 & 7 & 6 \\ 4 & 2 & 1 \end{pmatrix}$$

Für die Maschinenfolgematrix Q werden die Werte gewählt:

6. Testbeispiel:

$$Q = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 1 & 3 \\ 3 & 1 & 2 \\ 1 & 3 & 2 \end{pmatrix}$$

7. Testbeispiel:

$$Q = \begin{pmatrix} 2 & 3 & 1 \\ 3 & 2 & 1 \\ 2 & 1 & 3 \\ 3 & 2 & 1 \end{pmatrix}$$

8. Testbeispiel:

$$Q = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \\ 1 & 2 & 3 \\ 1 & 2 & 3 \end{pmatrix}$$

9. Testbeispiel:

$$Q = \begin{pmatrix} 3 & 1 & 2 \\ 3 & 2 & 1 \\ 1 & 2 & 3 \\ 1 & 2 & 3 \end{pmatrix}$$

10. Testbeispiel:

$$Q = \begin{pmatrix} 2 & 1 & 3 \\ 3 & 1 & 2 \\ 3 & 2 & 1 \\ 1 & 3 & 2 \end{pmatrix}$$

Mit diesen zehn Testbeispielen soll zunächst die „Minimierung der Zykluszeit" untersucht werden. Mit Hilfe des in Anhang 2 dokumentierten Modellgenerators wird das zu-

gehörige MANNE-Modell aufgebaut und mit dem Verfahren von BENDERS [9] in einer Modifikation von FLEISCHMANN [19] gelöst.

Die Zahl der Variablen beträgt 31 (davon 18 Binärvariablen), die Zahl der Restriktionen ist 48. Die numerischen Ergebnisse sind in einer Tabelle zusammengefaßt.

| Testbeispiel | Optimaler Zielfunktionswert in ZE | Gesamt CPU-Zeit in Sek. | Optimale Lösung gefunden nach ... Sek. | Rechenzeit für Beweisphase in Sek. | Zahl der Restriktionen im Bendersschen Unterprogramm |
|---|---|---|---|---|---|
| 1 | 26 | 43.5 | 3.0 | 40.5 | 51 |
| 2 | 30 | 6.8 | 3.8 | 3.0 | 29 |
| 3 | 29 | 210.7 | 4.5 | 206.2 | 38 |
| 4 | 28 | 28.6 | 3.2 | 25.4 | 36 |
| 5 | 30 | 13.9 | 4.5 | 9.4 | 35 |
| 6 | 22 | 126.0 | 3.6 | 122.4 | 33 |
| 7 | 27 | 177.3 | 9.6 | 167.7 | 68 |
| 8 | 27 | 380.4 | 255.1 | 125.3 | 216 |
| 9 | 22 | 6.2 | 4.3 | 1.9 | 25 |
| 10 | 21 | 3.3 | 3.0 | 0.3 | 10 |

Auffällig an den Ergebnissen für die Minimierung der Zykluszeit ist die Tatsache, daß bereits nach kurzer Zeit außer bei Testbeispiel 8 die optimale Lösung bekannt war. Bei Testbeispiel 8 wird allerdings schon nach 3 Sek. CPU-Zeit eine Lösung ermittelt, die nur 3 ZE schlechter als die gefundene optimale Lösung ist. Die hier gefundenen Ergebnisse bestätigen die Vermutung, die im 4. Kapitel beim Durchrechnen von Beispielen entstand, daß die Beweisphase die längste und aufwendigste Phase im Bendersschen Partitionsverfahren ist. Die Struktur der Koeffizienten-

matrix im Bendersschen Unterprogramm ist von besonderer Bedeutung. Werden nur zulässige Ablaufpläne konstruiert, d.h., ist das Netzwerk zyklenfrei, so stellt die Lösung des i-ten Bendersschen Unterprogramms auch eine zulässige Lösung des (i+1)-ten Bendersschen Unterprogramms dar, das sich nur um eine Zeile vom i-ten Programm unterscheidet. Tritt jedoch ein Zyklus auf, dann ist die i-te Lösung nicht mehr zulässig für das (i+1)-te Unterprogramm. Dadurch wird zur Lösung der „rein" ganzzahligen Unterprogramme erhebliche Rechenzeit verbraucht. Weiterhin beachtet werden muß der Besetztheitsgrad der Koeffizientenmatrix im Bendersschen Unterprogramm. Läuft der kritische Weg nicht über viele Auftragsfolgebedingungen, so treten dementsprechend wenige Binärvariablen im Bendersschen Unterprogramm auf.

Die Ergebnisse für die „Minimierung der Gesamtdurchlaufzeit" sind in der nächsten Tabelle aufgeführt. Die Anzahl der Variablen beträgt 30 (davon 18 Binärvariablen), die Anzahl der Restriktionen ist 44.

| Testbeispiel | Optimaler Zielfunktionswert in ZE | Gesamt CPU-Zeit in Sek. | Optimale Lösung gefunden nach ... Sek. | Rechenzeit für Beweisphase in Sek. | Zahl der Restriktionen im Bendersschen Unterprogramm |
|---|---|---|---|---|---|
| 1 | 80 | 103.2 | 1.8 | 101.4 | 84 |
| 2 | 95 | 138.3 | 3.1 | 135.2 | 105 |
| 3 | 93 | 131.0 | 1.2 | 129.8 | 121 |
| 4 | 84 | 157.8 | 3.8 | 154.0 | 67 |
| 5 | 100 | 74.6 | 26.1 | 48.5 | 95 |
| 6 | 71 | 487.8 | 318.0 | 169.8 | 74 |
| 7 | 74 | 460.0 | 390.9 | 69.1 | 147 |
| 8 | 79 | 311.2 | 258.5 | 52.7 | 309 |
| 9 | 73 | 234.4 | 116.1 | 118.3 | 123 |
| 10 | 67 | 107.0 | 61.2 | 45.8 | 118 |

Bei fast allen Testbeispielen ist ein Steigen der CPU-Zeit im Vergleich mit der CPU-Zeit bei der Minimierung der Zykluszeit zu beobachten. Dies ist hauptsächlich auf die Größe der zu lösenden Unterprogramme zurückzuführen. Bei der Minimierung der Zykluszeit ist die Anzahl der Restriktionen im Bendersschen Unterprogramm um das zwei- bis dreifache geringer als bei den Unterprogrammen bezüglich der Minimierung der Gesamtdurchlaufzeit. Außerdem ist die Struktur dieser Unterprogramme vollkommen verschieden. Bei der Minimierung der Zykluszeit haben die Koeffizienten der Binärvariablen stets den Wert UB, während in diesen Unterprogrammen die Größe UB mit unterschiedlichen Konstanten multipliziert wird.

Die Ergebnisse für die „Minimierung der Gesamtbelegungszeit" sind in der folgenden Tabelle aufgeführt. Die Anzahl der Variablen beträgt 33 (davon 18 Binärvariablen), die Anzahl der Restriktionen ist 56.

| Test-beispiel | Optimaler Zielfunktionswert in ZE | Gesamt CPU-Zeit in Sek. | Optimale Lösung gefunden nach ... Sek. | Rechenzeit für Beweisphase in Sek. | Zahl der Restriktionen im Bendersschen Unterprogramm |
|---|---|---|---|---|---|
| 1 | 71 | 323.2 | 2.1 | 321.1 | 71 |
| 2 | 84 | 124.0 | 4.6 | 119.4 | 55 |
| 3 | 76 | 73.8 | 8.2 | 65.6 | 96 |
| 4 | 75 | 451.5 | 4.8 | 446.7 | 72 |
| 5 | 83 | 128.1 | 2.7 | 125.4 | 70 |
| 6 | 64[1] | 600[1] | 13.2[1] | 586.8[1] | 121 |
| 7 | 69[1] | 600[1] | 15.1[1] | 584.9[1] | 111 |
| 8 | 70[1] | 600[1] | 319.9[1] | 280.1[1] | 102 |
| 9 | 63 | 239.9 | 3.6 | 236.3 | 110 |
| 10 | 57 | 74.2 | 8.5 | 75.7 | 81 |

1) Die Auftragsbearbeitung der Testbeispiele 6, 7 und 8 wird nach 600 Sek. abgebrochen, ohne daß die Optimalität der Lösung bewiesen wird. Für Testbeispiel 6 kann eine untere Schranke für den optimalen Zielwert mit 59 ZE, für Testbeispiel 7 mit 60 ZE und für Testbeispiel 8 mit 63 ZE angegeben werden.

Auch mit dem in Anhang 1 dokumentierten Programm konnten trotz Erzeugung von jeweils 100 aktiven (unverzögerten) Plänen keine besseren Zielwerte gefunden werden. Es liegt daher die Vermutung nahe, daß die aufgeführten Zielwerte die optimalen Werte darstellen.

Vergleicht man die CPU-Zeiten für die drei beschriebenen Zielsetzungen, so sieht man, daß die CPU-Zeit bei der Minimierung der Zykluszeit in den meisten Testbeispielen geringer ist als bei der Minimierung der Durchlaufzeit und der Belegungszeit. Dies ist auf die Netzwerkstruktur und die damit verbundene Struktur der Bendersschen Unterprogramme zurückzuführen. Auch die Zahl der Restriktionen für das „rein" ganzzahlige Unterprogramm ist bei der Minimierung der Zykluszeit in allen Fällen geringer als die Zahl der Restriktionen bei den übrigen Zielsetzungen.

Das Benderssche Partitionsverfahren wird an einem weiteren größeren Testbeispiel untersucht.

11. Testbeispiel (7 Aufträge, 5 Maschinen):

$$P = \begin{pmatrix} 1 & 8 & 4 & 3 & 9 \\ 5 & 8 & 8 & 10 & 6 \\ 9 & 6 & 9 & 9 & 4 \\ 6 & 6 & 5 & 2 & 6 \\ 4 & 10 & 10 & 9 & 4 \\ 9 & 10 & 1 & 7 & 4 \\ 9 & 5 & 6 & 7 & 2 \end{pmatrix} \qquad Q = \begin{pmatrix} 4 & 3 & 2 & 1 & 5 \\ 3 & 5 & 1 & 2 & 4 \\ 2 & 3 & 5 & 1 & 4 \\ 4 & 2 & 3 & 5 & 1 \\ 4 & 5 & 1 & 2 & 3 \\ 2 & 3 & 5 & 4 & 1 \\ 2 & 4 & 5 & 3 & 1 \end{pmatrix}$$

Es ergeben sich für die drei behandelten Zielsetzungen folgende Ergebnisse:

| Zielkriterium | Zielwert in ZE | Gesamt CPU-Zeit | Gefunden in |
|---|---|---|---|
| Zykluszeit Z | 66 | 900 Sek. | 224.3 Sek. |
| Gesamtbelegungszeit GB | 343 | 900 Sek. | 116.8 Sek. |
| Gesamtdurchlaufzeit GD | 325 | 900 Sek. | 97.0 Sek. |

Die Auftragsbearbeitung wird jeweils nach 900 Sekunden CPU-Zeit abgebrochen. Die Qualität der erzeugten Ergebnisse wird in Abschnitt 6.2 beurteilt.

## 6.2. Numerische Ergebnisse mit semiaktiven, aktiven und unverzögerten Ablaufplänen

Im Anhang 1 ist ein interaktives Programm dokumentiert, das wahlweise semiaktive, aktive oder unverzögerte Ablaufpläne erzeugt und das zur heuristischen Lösung des Job-Shop-Problems eingesetzt wird. Im Programm werden zuerst mit der Theorie, die im 3. Kapitel hergeleitet wurde, untere Schranken für die drei behandelten Zielsetzungen errechnet. Der Optimierungsprozeß läuft dann solange, bis die ermittelte Lösung nur noch um einen gewissen vorzugebenden Prozentsatz von der unteren Schranke abweicht, oder bis eine vorgegebene Zahl von Ablaufplänen errechnet wurde.

Um zu testen, welche von den drei verschiedenen Arten von Ablaufplänen die „besten" numerischen Ergebnisse bringen, werden für Testbeispiel 11 jeweils 100 semiaktive, aktive und unverzögerte Ablaufpläne erzeugt. In den folgenden Tabellen wird die Güte der erzeugten Ablaufpläne untersucht. Dabei wird unter Güte der folgende Quotient verstanden.

$$\text{Güte} := \frac{\text{Wert der Zielfunktion} - \text{Wert der unteren Schranke}}{\text{Wert der unteren Schranke}}$$

Dieser Quotient gibt also die maximal mögliche prozentuale Abweichung vom wahren unbekannten Zielwert an. In den nachstehenden Tabellen ist in jeder Spalte die Anzahl der Pläne angegeben, die maximal um x Prozent vom wahren, unbekannten Zielfunktionswert abweichen.

Für Testbeispiel 11 ergaben sich die folgenden unteren Schranken:

Zykluszeit Z = 56 ZE,
Gesamtbelegungszeit GB = 242 ZE,
Gesamtdurchlaufzeit GD = 245 ZE.

Für die 100 erzeugten semiaktiven Ablaufpläne ergibt sich:

| Zielkriterium | Anzahl der Pläne, die bis zu x Prozent von der unteren Schranke abweichen | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 | 100 | 110 |
| Zykluszeit Z | 0 | 0 | 1 | 4 | 10 | 20 | 28 | 38 | 52 | 69 | 84 |
| Gesamtbelegungszeit GB | 0 | 0 | 1 | 8 | 19 | 31 | 41 | 61 | 75 | 91 | 98 |
| Gesamtdurchlaufzeit GD | 0 | 0 | 0 | 0 | 0 | 1 | 2 | 8 | 22 | 36 | 47 |

| 120 | 130 | 140 | 150 | 160 | 170 | 180 | Bester Zielfunktionswert |
|---|---|---|---|---|---|---|---|
| 93 | 97 | 97 | 98 | 99 | 100 | - | 70 ZE |
| 97 | 99 | 99 | 99 | 100 | - | - | 308 ZE |
| 97 | 78 | 90 | 94 | 98 | 98 | 100 | 334 ZE |

Bildet man die Differenzen zwischen zwei benachtbarten Spalten, so erhält man die genaue Anzahl der Pläne, die zwischen x und x +10 Prozent von der unteren Schranke abweichen. Dabei fällt auf, daß der beste erzeugte Ablaufplan für die Zielsetzung der Durchlaufzeitminimierung um fast 60 Prozent von der unteren Schranke abweicht. Dies ist allerdings darauf zurückzuführen, daß zur Bestimmung der unteren Schranke nur Algorithmus 7 verwendet wird, damit die Zeit zur Berechnung einer unteren Schranke nicht zu hoch wird.

Die Ergebnisse für die aktiven Ablaufpläne finden sich in der nächsten Tabelle.

| Zielkriterium | Anzahl der Pläne, die bis zu x Prozent von der unteren Schranke abweichen | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 | Bester Zielfunktionswert |
| Zykluszeit Z | 0 | 15 | 38 | 70 | 90 | 97 | 100 | - | - | 63 ZE |
| Gesamtbelegungszeit GB | 0 | 8 | 57 | 93 | 100 | - | - | - | - | 276 ZE |
| Gesamtdurchlaufzeit GD | 0 | 0 | 0 | 0 | 1 | 44 | 89 | 99 | 100 | 355 ZE |

Für die unverzögerten Ablaufpläne ergab sich folgendes Bild:

| Zielkriterium | Anzahl der Pläne, die bis zu x Prozent von der unteren Schranke abweichen | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 10 | 20 | 30 | 40 | 50 | 60 | 70 | Bester Zielfunktionswert |
| Zykluszeit Z | 0 | 19 | 60 | 84 | 90 | 95 | 100 | 63 ZE |
| Gesamtbelegungszeit GB | 0 | 18 | 76 | 90 | 99 | 100 | - | 279 ZE |
| Gesamtdurchlaufzeit GD | 0 | 0 | 0 | 0 | 18 | 85 | 100 | 350 ZE |

Das beste Ergebnis für die drei Zielsetzungen lautet also: Z = 63 ZE, GB = 276 ZE und GD = 350 ZE.

Vergleicht man die Ergebnisse in den drei Tabellen, so läßt sich eindeutig die Dominanzbeziehung der aktiven und unverzögerten Ablaufpläne gegenüber den semiaktiven Ablaufplänen feststellen. Das beste erzielte Ergebnis für die drei Zielsetzungen lautet: Z = 70 ZE, GD = 308 ZE und GB = 384 ZE. In den benötigten Rechenzeiten - für einen semiaktiven Ablaufplan wurden 0.315 Sek. CPU-Zeit, für einen aktiven Ablaufplan 0.32 Sek. CPU-Zeit und für unverzögerte Ablaufpläne 0.32 Sek. CPU-Zeit benötigt - drückt sich aus, daß für die Erzeugung von unterschied-

lichen Arten von Ablaufplänen kaum ein Unterschied in den CPU-Zeiten vorhanden ist. Gewichtigere Unterschiede lassen sich erst bei größeren Testbeispielen feststellen. Nicht ganz so einfach läßt sich eine Dominanzbeziehung zwischen aktiven und unverzögerten Ablaufplänen herleiten. So sind beispielsweise die besten Zielwerte für die Zykluszeit gleich. Für die Gesamtdurchlaufzeit liefern die unverzögerten Ablaufpläne die beste Lösung, während für die Minimierung der Belegungszeit die aktiven Pläne den besten Zielwert liefern. Vergleicht man die Anzahl der Pläne, die jeweils um eine gewisse Prozentzahl nur von der unteren Schranke abweichen, so kann festgestellt werden, daß die unverzögerten Ablaufpläne bessere Ergebnisse liefern als die aktiven Ablaufpläne. Umfangreiche Untersuchungen von CONWAY/MAXWELL/MILLER [15] haben dies 1960 bereits ergeben.

Vergleicht man diese Resultate mit denen, die mit der Bendersschen Partitionsmethode erzielt wurden, so müssen die zwei Komponenten - Güte der gefundenen Lösung und Rechenzeitverbrauch - nebeneinander gestellt werden. Für die Zykluszeitminimierung ist das Ergebnis, das mit den aktiven (unverzögerten) Ablaufplänen gefunden wurde, um 3 ZE besser als das mit der Bendersschen Partitionsmethode gefundene Ergebnis. Die Benderssche Partitionsmethode produziert allerdings mit einem Ergebnis von 343 ZE für die Gesamtdurchlaufzeit einen um 7 ZE besseren Ablaufplan. Für die Belegungszeit kann die Benderssche Partitionsmethode nach 900 Sek. nur einen Ablaufplan erzeugen, der um 49 ZE schlechter ist als der beste unverzögerte Ablaufplan.

Beim Vergleich der CPU-Zeiten fällt eine eindeutige Dominanz der simulativen Verfahren auf. Dies bedeutet jedoch nicht, daß die Benderssche Methode vollkommen ungeeignet für die Lösung von Scheduling-Problemen ist.

Die meiste Rechenzeit wird zur Lösung der „rein" ganzzahligen Unterprogramme verwendet. Gerade auf dem Gebiet der ganzzahligen Programmierung ist man sehr bemüht, neue leistungsfähige Algorithmen zu entwickeln (vgl. z.B. SALKIN [53]). Es besteht kein Zweifel daran, daß mit fortschreitender Entwicklung des Algorithmus die Benderssche Methode auch zur Lösung von größeren Ablaufplanungsproblemen eingesetzt werden kann. Es kann aber kein Zweifel daran bestehen, daß durch die NP-Vollständigkeit der ganzzahligen Programmierung die Größenordnungsverhältnisse so gering sein werden, daß exakte Optimierungsverfahren für sehr große Ablaufplanungsprobleme nicht in Frage kommen.

Um zu erproben, wie gut man auch größere Testbeispiele simulieren kann, werden an zwei großen Testbeispielen mit Hilfe des Programms aus Anhang 1 Simulationsläufe gestartet. In der Literatur über Ablaufplanungsprobleme sind kaum größere Testbeispiele zu finden, für die Lösungen bekannt sind. Deshalb werden die Matrizen P und Q hier abgedruckt und die ermittelten Ergebnisse wiedergegeben.

12. Testbeispiel (20 Aufträge, 20 Maschinen):
Für dieses Beispiel werden 20 unverzögerte Ablaufpläne erzeugt. In der nächsten Tabelle sind die gefundenen Zielwerte aufgeführt.

Bearbeitungsdauer P[J,M] des Auftrags J auf Maschine M:

| J \ M | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 7 | 6 | 8 | 1 | 8 | 2 | 9 | 1 | 6 | 6 | 1 | 10 | 6 | 7 | 4 | 10 | 4 | 2 | 8 |
| 2 | 5 | 3 | 2 | 4 | 10 | 5 | 2 | 1 | 3 | 3 | 5 | 7 | 5 | 4 | 1 | 2 | 4 | 5 | 8 | 9 |
| 3 | 5 | 10 | 2 | 10 | 6 | 10 | 9 | 3 | 5 | 10 | 5 | 7 | 8 | 8 | 10 | 9 | 8 | 9 | 5 | 4 |
| 4 | 7 | 7 | 3 | 4 | 4 | 9 | 8 | 3 | 5 | 2 | 9 | 5 | 2 | 4 | 3 | 6 | 9 | 9 | 2 | 1 |
| 5 | 6 | 3 | 9 | 2 | 1 | 8 | 6 | 1 | 3 | 9 | 4 | 8 | 6 | 10 | 5 | 1 | 9 | 3 | 4 | 10 |
| 6 | 2 | 5 | 4 | 9 | 1 | 8 | 9 | 6 | 4 | 7 | 3 | 2 | 9 | 2 | 8 | 6 | 10 | 2 | 3 | 7 |
| 7 | 6 | 4 | 5 | 10 | 7 | 9 | 1 | 7 | 2 | 7 | 7 | 5 | 8 | 2 | 6 | 10 | 1 | 3 | 9 | 5 |
| 8 | 2 | 3 | 2 | 2 | 4 | 10 | 6 | 3 | 1 | 3 | 5 | 2 | 8 | 7 | 2 | 9 | 6 | 10 | 9 | 5 |
| 9 | 3 | 3 | 5 | 2 | 10 | 4 | 5 | 5 | 3 | 7 | 5 | 9 | 3 | 7 | 6 | 6 | 6 | 1 | 3 | 6 |
| 10 | 8 | 9 | 5 | 3 | 4 | 8 | 3 | 9 | 5 | 4 | 1 | 5 | 3 | 7 | 4 | 3 | 9 | 4 | 9 | 3 |
| 11 | 6 | 6 | 3 | 8 | 7 | 3 | 6 | 2 | 4 | 6 | 6 | 8 | 6 | 3 | 5 | 1 | 3 | 6 | 7 | 4 |
| 12 | 6 | 8 | 8 | 2 | 7 | 10 | 4 | 2 | 1 | 8 | 6 | 1 | 8 | 5 | 9 | 6 | 5 | 8 | 8 | 1 |
| 13 | 9 | 2 | 9 | 2 | 2 | 2 | 7 | 3 | 5 | 1 | 7 | 2 | 5 | 2 | 6 | 3 | 9 | 9 | 4 | 10 |
| 14 | 5 | 10 | 1 | 5 | 8 | 10 | 7 | 4 | 9 | 5 | 1 | 4 | 2 | 10 | 8 | 6 | 4 | 10 | 3 | 6 |
| 15 | 1 | 4 | 4 | 9 | 8 | 7 | 5 | 8 | 7 | 2 | 6 | 9 | 5 | 3 | 6 | 2 | 1 | 7 | 7 | 5 |
| 16 | 5 | 3 | 9 | 2 | 7 | 9 | 10 | 7 | 4 | 10 | 10 | 10 | 9 | 6 | 7 | 4 | 9 | 1 | 7 | 7 |
| 17 | 6 | 10 | 10 | 1 | 2 | 7 | 1 | 9 | 2 | 2 | 5 | 1 | 1 | 4 | 8 | 8 | 5 | 5 | 2 | 5 |
| 18 | 2 | 7 | 8 | 9 | 2 | 2 | 2 | 5 | 3 | 4 | 5 | 5 | 5 | 9 | 2 | 3 | 8 | 2 | 8 | 9 |
| 19 | 5 | 10 | 5 | 2 | 1 | 2 | 7 | 5 | 8 | 3 | 10 | 6 | 5 | 4 | 10 | 1 | 5 | 6 | 3 | 2 |
| 20 | 3 | 1 | 7 | 5 | 9 | 9 | 2 | 8 | 5 | 6 | 10 | 7 | 2 | 5 | 6 | 9 | 5 | 6 | 5 | 8 |

Maschinenfolgematrix Q[J,M]:

| J \ M | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 7 | 3 | 18 | 10 | 4 | 1 | 17 | 19 | 9 | 16 | 13 | 14 | 8 | 5 | 11 | 20 | 2 | 12 | 15 | 6 |
| 2 | 10 | 3 | 9 | 5 | 8 | 4 | 6 | 14 | 18 | 16 | 11 | 13 | 20 | 12 | 2 | 1 | 15 | 7 | 17 | 19 |
| 3 | 8 | 17 | 18 | 9 | 19 | 10 | 5 | 16 | 6 | 20 | 14 | 3 | 7 | 13 | 2 | 4 | 1 | 11 | 15 | 12 |
| 4 | 4 | 14 | 16 | 9 | 6 | 10 | 17 | 8 | 19 | 15 | 13 | 7 | 11 | 18 | 1 | 5 | 3 | 12 | 2 | 20 |
| 5 | 7 | 16 | 15 | 10 | 14 | 3 | 19 | 17 | 12 | 4 | 11 | 2 | 20 | 1 | 6 | 18 | 9 | 5 | 8 | 13 |
| 6 | 5 | 7 | 8 | 4 | 9 | 17 | 1 | 2 | 15 | 19 | 13 | 3 | 14 | 16 | 12 | 11 | 20 | 18 | 10 | 6 |
| 7 | 12 | 9 | 1 | 16 | 2 | 7 | 10 | 3 | 6 | 5 | 19 | 8 | 15 | 11 | 13 | 18 | 20 | 4 | 17 | 14 |
| 8 | 8 | 7 | 17 | 2 | 9 | 15 | 1 | 6 | 20 | 16 | 13 | 10 | 3 | 12 | 18 | 14 | 11 | 5 | 19 | 4 |
| 9 | 3 | 4 | 5 | 15 | 8 | 18 | 1 | 13 | 11 | 2 | 7 | 14 | 20 | 17 | 19 | 10 | 12 | 16 | 6 | 9 |
| 10 | 19 | 8 | 6 | 12 | 13 | 18 | 4 | 2 | 5 | 16 | 14 | 17 | 15 | 7 | 9 | 11 | 10 | 3 | 20 | 1 |
| 11 | 8 | 9 | 5 | 7 | 13 | 14 | 3 | 17 | 20 | 16 | 15 | 19 | 12 | 2 | 10 | 18 | 4 | 11 | 1 | 6 |
| 12 | 3 | 2 | 6 | 19 | 7 | 9 | 12 | 13 | 18 | 11 | 14 | 1 | 10 | 20 | 15 | 5 | 4 | 16 | 17 | 8 |
| 13 | 15 | 14 | 4 | 20 | 7 | 19 | 3 | 13 | 2 | 11 | 6 | 5 | 18 | 17 | 10 | 9 | 16 | 1 | 8 | 12 |
| 14 | 19 | 17 | 5 | 15 | 3 | 4 | 14 | 7 | 1 | 20 | 12 | 9 | 13 | 6 | 18 | 2 | 8 | 16 | 10 | 11 |
| 15 | 8 | 1 | 13 | 18 | 11 | 10 | 20 | 2 | 17 | 5 | 15 | 12 | 19 | 9 | 3 | 14 | 6 | 16 | 4 | 7 |
| 16 | 12 | 19 | 7 | 17 | 4 | 18 | 8 | 11 | 16 | 3 | 10 | 9 | 5 | 15 | 2 | 6 | 20 | 1 | 14 | 13 |
| 17 | 7 | 3 | 18 | 16 | 20 | 9 | 4 | 11 | 1 | 5 | 15 | 14 | 19 | 10 | 13 | 6 | 2 | 17 | 12 | 8 |
| 18 | 10 | 7 | 13 | 4 | 5 | 3 | 12 | 6 | 20 | 15 | 19 | 1 | 16 | 9 | 2 | 8 | 17 | 18 | 11 | 14 |
| 19 | 3 | 4 | 12 | 2 | 11 | 13 | 9 | 18 | 16 | 15 | 20 | 10 | 7 | 17 | 6 | 1 | 19 | 5 | 14 | 8 |
| 20 | 5 | 15 | 8 | 9 | 6 | 19 | 12 | 10 | 7 | 4 | 16 | 14 | 3 | 17 | 1 | 2 | 18 | 13 | 11 | 20 |

| Plan-Nr. | Zykluszeit | Gesamtbelegungs-zeit | Gesamtdurch-laufzeit |
|---|---|---|---|
| 1 | 227 | 3934 | 3475 |
| 2 | 225 | 3918 | 3632 |
| 3 | 215 | 3711 | 3518 |
| 4 | 213 | 3844 | 3514 |
| 5 | 204 | 3534 | 3333 |
| 6 | 209 | 3752 | 3497 |
| 7 | 218 | 3755 | 3478 |
| 8 | 218 | 3792 | 3454 |
| 9 | 206 | 3726 | 3517 |
| 10 | 226 | 3885 | 3514 |
| 11 | 224 | 3904 | 3679 |
| 12 | 216 | 3860 | 3432 |
| 13 | 222 | 3901 | 3641 |
| 14 | 217 | 3786 | 3582 |
| 15 | 219 | 3726 | 3467 |
| 16 | 220 | 3861 | 3534 |
| 17 | 222 | 3789 | 3521 |
| 18 | 211 | 3783 | 3577 |
| 19 | 202 | 3664 | 3581 |
| 20 | 224 | 3725 | 3521 |

Den besten Zielwert für die Zykluszeit liefert Ablaufplan 19 mit Z = 202 ZE, für die Gesamtbelegungszeit Ablaufplan 5 mit GB = 3534 ZE und für die Gesamtdurchlaufzeit Ablaufplan 5 mit GD = 3333 ZE. Die untere Schranke für die Zykluszeit ist Z = 156 ZE, für die Gesamtbelegungszeit GB = 2479 ZE und für die Gesamtdurchlaufzeit GD = 2301 ZE. Dies ergibt also eine maximale Abweichung für die Zykluszeit um 29.4 Prozent, für die Gesamtbelegungszeit um 42.5 Prozent und für die Gesamtdurchlaufzeit um 43.1 Prozent vom optimalen, unbekannten Zielwert. Die benötigte Rechenzeit zur Erzeugung eines unverzögerten Ablaufplans beträgt 15.5 Sek.

13. Testbeispiel (50 Aufträge, 20 Maschinen):
Für dieses Beispiel sind die Matrizen P und Q auf den nächsten Seiten aufgeführt. In der folgenden Tabelle sind die Zielwerte für 20 unverzögerte Ablaufpläne aufgeführt.

| Plan-Nr. | Zykluszeit | Gesamtbelegungs-zeit | Gesamtdurch-laufzeit |
|---|---|---|---|
| 1 | 412 | 7468 | 15723 |
| 2 | 413 | 7727 | 16240 |
| 3 | 448 | 8066 | 15884 |
| 4 | 408 | 7531 | 15847 |
| 5 | 413 | 7646 | 15996 |
| 6 | 417 | 7811 | 16210 |
| 7 | 444 | 7943 | 16092 |
| 8 | 423 | 7551 | 15809 |
| 9 | 399 | 7401 | 15681 |
| 10 | 391 | 7361 | 15482 |
| 11 | 428 | 7559 | 15763 |
| 12 | 413 | 7685 | 16434 |
| 13 | 397 | 7447 | 15834 |
| 14 | 405 | 7428 | 15697 |
| 15 | 446 | 7901 | 15695 |
| 16 | 406 | 7472 | 15618 |
| 17 | 403 | 7609 | 15832 |
| 18 | 405 | 7483 | 15730 |
| 19 | 435 | 8012 | 16104 |
| 20 | 419 | 7913 | 16151 |

Den besten Zielwert mit Z = 391 ZE für die Zykluszeit liefert Ablaufplan 10, für die Gesamtbelegungszeit GB = 7361 ZE Ablaufplan 10 und für die Gesamtdurchlaufzeit mit GD = 15482 Ablaufplan 10.

Für die Zykluszeit ergibt sich eine untere Schranke von 311 ZE, für die Durchlaufzeit von 5623 ZE und für die Belegungszeit von 5551 ZE. Dies bedeutet für die Zykluszeit eine maximale Abweichung von 25.7 Prozent, für die Durchlaufzeit von 177 Prozent und für die Belegungszeit um 32.4 Prozent vom optimalen unbekannten Zielwert. Zur Erzeugung eines unverzögerten Ablaufplans wurden 103.2 Sek. CPU-Zeit benötigt.

Bearbeitungsdauer P[J,M] des Auftrags J auf Maschine M:

| J \ M | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 7 | 6 | 8 | 1 | 8 | 2 | 9 | 1 | 6 | 6 | 1 | 10 | 6 | 7 | 4 | 10 | 4 | 2 | 8 |
| 2 | 5 | 3 | 2 | 4 | 10 | 5 | 2 | 1 | 3 | 3 | 5 | 7 | 5 | 4 | 1 | 2 | 4 | 5 | 8 | 9 |
| 3 | 5 | 10 | 2 | 10 | 6 | 10 | 9 | 3 | 5 | 10 | 5 | 7 | 8 | 8 | 10 | 9 | 8 | 9 | 5 | 4 |
| 4 | 7 | 7 | 3 | 4 | 4 | 9 | 8 | 3 | 5 | 2 | 9 | 5 | 2 | 4 | 3 | 6 | 9 | 9 | 2 | 1 |
| 5 | 6 | 3 | 9 | 2 | 1 | 8 | 6 | 1 | 3 | 9 | 4 | 8 | 6 | 10 | 5 | 1 | 9 | 3 | 4 | 10 |
| 6 | 2 | 5 | 4 | 9 | 1 | 8 | 9 | 6 | 4 | 7 | 3 | 2 | 9 | 2 | 8 | 6 | 10 | 2 | 3 | 7 |
| 7 | 6 | 4 | 5 | 10 | 7 | 9 | 1 | 7 | 2 | 7 | 7 | 5 | 8 | 2 | 6 | 10 | 1 | 3 | 9 | 5 |
| 8 | 2 | 3 | 2 | 2 | 4 | 10 | 6 | 3 | 1 | 3 | 5 | 2 | 8 | 7 | 2 | 9 | 6 | 10 | 9 | 5 |
| 9 | 3 | 3 | 5 | 2 | 10 | 4 | 5 | 5 | 3 | 7 | 5 | 9 | 3 | 7 | 6 | 6 | 6 | 1 | 3 | 6 |
| 10 | 8 | 9 | 5 | 3 | 4 | 8 | 3 | 9 | 5 | 4 | 1 | 5 | 3 | 7 | 4 | 3 | 9 | 4 | 9 | 3 |
| 11 | 6 | 6 | 3 | 8 | 7 | 3 | 6 | 2 | 4 | 6 | 6 | 8 | 6 | 3 | 5 | 1 | 3 | 6 | 7 | 4 |
| 12 | 6 | 8 | 8 | 2 | 7 | 10 | 4 | 2 | 1 | 8 | 6 | 1 | 8 | 5 | 9 | 6 | 5 | 8 | 8 | 1 |
| 13 | 9 | 2 | 9 | 2 | 2 | 2 | 7 | 3 | 5 | 1 | 7 | 2 | 5 | 2 | 6 | 3 | 9 | 9 | 4 | 10 |
| 14 | 5 | 10 | 1 | 5 | 8 | 10 | 7 | 4 | 9 | 5 | 1 | 4 | 2 | 10 | 8 | 6 | 4 | 10 | 3 | 6 |
| 15 | 1 | 4 | 4 | 9 | 8 | 7 | 5 | 8 | 7 | 2 | 6 | 9 | 5 | 3 | 6 | 2 | 1 | 7 | 7 | 5 |
| 16 | 5 | 3 | 9 | 2 | 7 | 9 | 10 | 7 | 4 | 10 | 10 | 10 | 9 | 6 | 7 | 4 | 9 | 1 | 7 | 7 |
| 17 | 6 | 10 | 10 | 1 | 2 | 7 | 1 | 9 | 2 | 2 | 5 | 1 | 1 | 4 | 8 | 8 | 5 | 5 | 2 | 5 |
| 18 | 2 | 7 | 8 | 9 | 2 | 2 | 2 | 5 | 3 | 4 | 5 | 5 | 5 | 9 | 2 | 3 | 8 | 2 | 8 | 9 |
| 19 | 5 | 10 | 5 | 2 | 1 | 2 | 7 | 5 | 8 | 3 | 10 | 6 | 5 | 4 | 10 | 1 | 5 | 6 | 3 | 2 |
| 20 | 3 | 1 | 7 | 5 | 9 | 9 | 2 | 8 | 5 | 6 | 10 | 7 | 2 | 5 | 6 | 9 | 5 | 6 | 5 | 8 |
| 21 | 9 | 2 | 10 | 9 | 4 | 1 | 10 | 6 | 6 | 9 | 3 | 2 | 6 | 6 | 1 | 6 | 8 | 10 | 7 | 3 |
| 22 | 3 | 4 | 9 | 6 | 8 | 6 | 2 | 8 | 9 | 10 | 4 | 4 | 4 | 5 | 3 | 5 | 1 | 5 | 1 | 4 |
| 23 | 1 | 3 | 5 | 2 | 7 | 5 | 3 | 10 | 8 | 7 | 1 | 1 | 3 | 5 | 1 | 7 | 10 | 9 | 8 | 5 |
| 24 | 4 | 2 | 2 | 10 | 3 | 10 | 5 | 7 | 10 | 8 | 4 | 2 | 5 | 10 | 5 | 10 | 7 | 4 | 5 | 2 |
| 25 | 4 | 4 | 9 | 5 | 3 | 4 | 10 | 1 | 3 | 9 | 4 | 5 | 1 | 9 | 10 | 10 | 6 | 3 | 7 | 4 |
| 26 | 7 | 6 | 4 | 8 | 10 | 1 | 10 | 8 | 8 | 1 | 3 | 4 | 5 | 4 | 6 | 4 | 7 | 2 | 5 | 3 |
| 27 | 9 | 8 | 8 | 1 | 7 | 8 | 8 | 5 | 9 | 8 | 4 | 10 | 3 | 1 | 4 | 9 | 1 | 1 | 7 | 1 |
| 28 | 7 | 9 | 3 | 1 | 1 | 7 | 10 | 10 | 9 | 4 | 2 | 3 | 4 | 2 | 9 | 4 | 9 | 3 | 1 | 6 |
| 29 | 7 | 6 | 10 | 10 | 10 | 1 | 6 | 8 | 1 | 4 | 6 | 3 | 6 | 5 | 8 | 1 | 5 | 8 | 4 | 5 |
| 30 | 3 | 8 | 3 | 8 | 8 | 10 | 4 | 10 | 4 | 3 | 4 | 9 | 9 | 5 | 4 | 5 | 5 | 8 | 10 | 3 |
| 31 | 1 | 2 | 5 | 1 | 9 | 1 | 8 | 10 | 9 | 2 | 1 | 3 | 8 | 7 | 4 | 4 | 5 | 2 | 3 | 4 |
| 32 | 6 | 10 | 10 | 3 | 4 | 9 | 9 | 1 | 8 | 5 | 6 | 7 | 7 | 6 | 4 | 2 | 1 | 7 | 9 | 5 |
| 33 | 2 | 7 | 8 | 10 | 5 | 6 | 10 | 2 | 3 | 4 | 7 | 1 | 8 | 10 | 10 | 3 | 4 | 4 | 8 | 10 |
| 34 | 4 | 2 | 7 | 5 | 8 | 3 | 4 | 1 | 3 | 1 | 1 | 3 | 3 | 5 | 7 | 1 | 10 | 2 | 10 | 4 |
| 35 | 10 | 8 | 4 | 1 | 6 | 2 | 1 | 10 | 5 | 8 | 9 | 5 | 9 | 3 | 5 | 2 | 7 | 9 | 5 | 8 |
| 36 | 2 | 4 | 7 | 10 | 5 | 4 | 10 | 8 | 5 | 6 | 8 | 10 | 1 | 2 | 4 | 8 | 10 | 9 | 10 | 2 |
| 37 | 8 | 2 | 10 | 8 | 5 | 3 | 10 | 1 | 9 | 3 | 8 | 1 | 3 | 4 | 4 | 10 | 2 | 10 | 4 | 9 |
| 38 | 4 | 3 | 7 | 8 | 7 | 6 | 8 | 3 | 3 | 10 | 1 | 7 | 8 | 1 | 5 | 4 | 3 | 2 | 5 | 6 |
| 39 | 3 | 8 | 10 | 10 | 1 | 4 | 5 | 1 | 7 | 6 | 8 | 9 | 8 | 1 | 6 | 6 | 4 | 7 | 1 | 6 |
| 40 | 2 | 2 | 4 | 6 | 10 | 9 | 7 | 9 | 9 | 8 | 1 | 9 | 2 | 4 | 10 | 5 | 3 | 2 | 9 | 1 |
| 41 | 1 | 5 | 1 | 7 | 2 | 3 | 9 | 4 | 2 | 4 | 7 | 8 | 7 | 1 | 6 | 10 | 9 | 10 | 2 | 2 |
| 42 | 8 | 7 | 4 | 8 | 6 | 6 | 5 | 1 | 9 | 2 | 9 | 10 | 2 | 3 | 9 | 5 | 6 | 4 | 5 | 9 |
| 43 | 1 | 5 | 8 | 10 | 7 | 9 | 9 | 5 | 6 | 5 | 2 | 9 | 7 | 5 | 9 | 10 | 8 | 8 | 3 | 3 |
| 44 | 1 | 8 | 2 | 9 | 7 | 4 | 10 | 7 | 4 | 8 | 7 | 9 | 9 | 5 | 2 | 3 | 9 | 2 | 4 | 5 |
| 45 | 6 | 5 | 8 | 5 | 8 | 9 | 8 | 9 | 10 | 1 | 1 | 9 | 7 | 1 | 8 | 2 | 7 | 5 | 3 | 9 |
| 46 | 3 | 8 | 7 | 9 | 4 | 1 | 5 | 9 | 3 | 4 | 9 | 3 | 3 | 3 | 6 | 3 | 10 | 1 | 6 | 5 |
| 47 | 4 | 9 | 7 | 6 | 10 | 8 | 10 | 1 | 6 | 2 | 2 | 8 | 3 | 7 | 7 | 3 | 1 | 8 | 9 | 3 |
| 48 | 8 | 3 | 9 | 3 | 9 | 5 | 3 | 7 | 9 | 10 | 10 | 7 | 8 | 10 | 3 | 7 | 2 | 5 | 8 | 10 |
| 49 | 8 | 8 | 5 | 3 | 1 | 2 | 4 | 8 | 4 | 10 | 1 | 7 | 4 | 4 | 8 | 8 | 10 | 9 | 10 | 7 |
| 50 | 5 | 10 | 8 | 10 | 7 | 5 | 6 | 3 | 5 | 7 | 6 | 1 | 3 | 6 | 9 | 4 | 6 | 3 | 7 | 6 |

Maschinenfolgematrix Q[J,M]:

| J \ M | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 19 | 1 | 20 | 15 | 3 | 2 | 10 | 9 | 4 | 14 | 11 | 7 | 13 | 17 | 8 | 6 | 16 | 18 | 5 | 12 |
| 2 | 16 | 17 | 3 | 7 | 18 | 20 | 4 | 5 | 2 | 10 | 8 | 9 | 6 | 19 | 13 | 11 | 1 | 14 | 12 | 15 |
| 3 | 9 | 16 | 18 | 7 | 14 | 15 | 13 | 11 | 1 | 12 | 5 | 3 | 6 | 4 | 19 | 20 | 17 | 2 | 8 | 10 |
| 4 | 6 | 12 | 3 | 17 | 16 | 7 | 4 | 13 | 10 | 2 | 20 | 18 | 8 | 5 | 9 | 14 | 1 | 11 | 19 | 15 |
| 5 | 12 | 15 | 10 | 5 | 19 | 14 | 7 | 17 | 8 | 13 | 1 | 6 | 16 | 18 | 2 | 4 | 9 | 20 | 3 | 11 |
| 6 | 15 | 20 | 16 | 8 | 14 | 2 | 7 | 3 | 1 | 18 | 10 | 19 | 11 | 13 | 9 | 5 | 17 | 12 | 6 | 4 |
| 7 | 2 | 1 | 13 | 9 | 5 | 18 | 7 | 20 | 14 | 17 | 16 | 11 | 12 | 4 | 10 | 8 | 15 | 6 | 19 | 3 |
| 8 | 7 | 14 | 6 | 19 | 2 | 13 | 3 | 9 | 4 | 12 | 1 | 18 | 16 | 15 | 8 | 10 | 5 | 17 | 11 | 20 |
| 9 | 18 | 8 | 20 | 9 | 1 | 7 | 4 | 16 | 15 | 11 | 14 | 19 | 12 | 2 | 13 | 17 | 3 | 10 | 6 | 5 |
| 10 | 18 | 20 | 3 | 12 | 17 | 8 | 4 | 13 | 9 | 7 | 5 | 19 | 6 | 1 | 16 | 15 | 2 | 11 | 14 | 10 |
| 11 | 20 | 19 | 3 | 14 | 5 | 16 | 11 | 8 | 1 | 12 | 4 | 13 | 17 | 9 | 7 | 10 | 18 | 2 | 15 | 6 |
| 12 | 1 | 18 | 12 | 4 | 16 | 10 | 9 | 14 | 19 | 7 | 5 | 17 | 11 | 6 | 2 | 13 | 8 | 3 | 20 | 15 |
| 13 | 16 | 6 | 18 | 19 | 11 | 7 | 14 | 5 | 10 | 8 | 17 | 9 | 15 | 2 | 13 | 4 | 1 | 3 | 20 | 12 |
| 14 | 8 | 11 | 18 | 2 | 17 | 12 | 5 | 4 | 14 | 19 | 13 | 15 | 6 | 7 | 3 | 20 | 10 | 9 | 1 | 16 |
| 15 | 16 | 9 | 7 | 8 | 17 | 4 | 19 | 6 | 12 | 10 | 13 | 2 | 18 | 15 | 5 | 14 | 3 | 20 | 11 | 1 |
| 16 | 12 | 11 | 20 | 2 | 13 | 4 | 10 | 6 | 19 | 18 | 3 | 9 | 16 | 1 | 5 | 14 | 8 | 7 | 15 | 17 |
| 17 | 2 | 14 | 1 | 15 | 16 | 5 | 10 | 18 | 7 | 8 | 11 | 9 | 20 | 4 | 13 | 19 | 6 | 3 | 12 | 17 |
| 18 | 9 | 3 | 16 | 13 | 19 | 11 | 20 | 15 | 4 | 12 | 7 | 2 | 17 | 10 | 14 | 5 | 8 | 1 | 18 | 6 |
| 19 | 10 | 17 | 6 | 14 | 9 | 16 | 13 | 15 | 20 | 7 | 1 | 3 | 18 | 4 | 19 | 2 | 11 | 8 | 12 | 5 |
| 20 | 15 | 19 | 14 | 11 | 20 | 6 | 16 | 9 | 2 | 12 | 1 | 8 | 7 | 5 | 10 | 4 | 17 | 3 | 18 | 13 |
| 21 | 20 | 6 | 17 | 18 | 3 | 8 | 19 | 5 | 16 | 4 | 14 | 9 | 2 | 10 | 11 | 12 | 7 | 15 | 1 | 13 |
| 22 | 13 | 20 | 17 | 16 | 11 | 12 | 3 | 19 | 1 | 9 | 4 | 2 | 18 | 14 | 10 | 5 | 6 | 15 | 7 | 8 |
| 23 | 1 | 11 | 6 | 14 | 4 | 12 | 16 | 13 | 2 | 18 | 5 | 3 | 7 | 10 | 9 | 19 | 15 | 6 | 20 | 17 |
| 24 | 7 | 10 | 18 | 9 | 5 | 4 | 13 | 16 | 2 | 19 | 12 | 3 | 6 | 8 | 1 | 15 | 17 | 14 | 11 | 20 |
| 25 | 14 | 18 | 7 | 17 | 1 | 4 | 5 | 19 | 12 | 11 | 10 | 9 | 6 | 2 | 16 | 3 | 15 | 8 | 20 | 13 |
| 26 | 6 | 12 | 10 | 9 | 19 | 4 | 7 | 17 | 8 | 15 | 18 | 11 | 1 | 3 | 14 | 16 | 5 | 13 | 2 | 20 |
| 27 | 7 | 3 | 9 | 2 | 14 | 11 | 13 | 8 | 6 | 20 | 1 | 17 | 16 | 15 | 4 | 16 | 12 | 19 | 10 | 5 |
| 28 | 3 | 9 | 15 | 17 | 4 | 7 | 16 | 19 | 8 | 11 | 5 | 2 | 1 | 14 | 20 | 12 | 6 | 10 | 13 | 18 |
| 29 | 13 | 2 | 10 | 4 | 18 | 20 | 5 | 7 | 9 | 1 | 14 | 3 | 11 | 12 | 15 | 8 | 6 | 19 | 17 | 16 |
| 30 | 9 | 14 | 2 | 18 | 20 | 4 | 3 | 16 | 1 | 7 | 15 | 10 | 8 | 12 | 11 | 6 | 17 | 13 | 5 | 19 |
| 31 | 13 | 15 | 10 | 6 | 7 | 11 | 4 | 9 | 18 | 14 | 17 | 2 | 3 | 12 | 19 | 1 | 8 | 16 | 20 | 5 |
| 32 | 3 | 17 | 12 | 13 | 10 | 7 | 18 | 14 | 9 | 20 | 6 | 16 | 19 | 11 | 4 | 8 | 15 | 2 | 5 | 1 |
| 33 | 1 | 11 | 10 | 7 | 14 | 2 | 8 | 15 | 18 | 3 | 12 | 19 | 13 | 4 | 9 | 16 | 17 | 20 | 5 | 6 |
| 34 | 20 | 10 | 6 | 18 | 19 | 9 | 7 | 1 | 11 | 5 | 12 | 16 | 4 | 15 | 8 | 2 | 17 | 14 | 13 | 3 |
| 35 | 20 | 13 | 5 | 15 | 10 | 17 | 16 | 4 | 3 | 19 | 18 | 1 | 2 | 14 | 9 | 6 | 7 | 12 | 8 | 11 |
| 36 | 10 | 2 | 17 | 6 | 19 | 18 | 3 | 5 | 9 | 13 | 4 | 14 | 15 | 11 | 20 | 12 | 8 | 7 | 1 | 16 |
| 37 | 7 | 2 | 1 | 10 | 3 | 6 | 19 | 18 | 8 | 5 | 13 | 4 | 15 | 16 | 12 | 17 | 11 | 9 | 14 | 20 |
| 38 | 7 | 13 | 5 | 19 | 14 | 9 | 20 | 12 | 15 | 10 | 18 | 8 | 4 | 11 | 2 | 16 | 6 | 17 | 3 | 1 |
| 39 | 4 | 15 | 12 | 13 | 19 | 16 | 7 | 1 | 9 | 18 | 6 | 5 | 8 | 11 | 17 | 2 | 10 | 3 | 14 | 20 |
| 40 | 16 | 12 | 4 | 2 | 18 | 15 | 5 | 10 | 7 | 1 | 8 | 11 | 14 | 9 | 13 | 6 | 19 | 3 | 20 | 17 |
| 41 | 9 | 7 | 19 | 11 | 8 | 3 | 15 | 6 | 17 | 18 | 14 | 10 | 2 | 20 | 13 | 12 | 5 | 1 | 16 | 4 |
| 42 | 11 | 2 | 12 | 1 | 4 | 3 | 18 | 16 | 19 | 14 | 5 | 17 | 10 | 6 | 15 | 13 | 8 | 20 | 9 | 7 |
| 43 | 15 | 4 | 13 | 12 | 3 | 16 | 11 | 5 | 1 | 20 | 7 | 14 | 9 | 8 | 17 | 19 | 2 | 10 | 18 | 6 |
| 44 | 20 | 7 | 9 | 14 | 8 | 11 | 4 | 17 | 12 | 16 | 18 | 10 | 13 | 19 | 15 | 2 | 5 | 1 | 3 | 6 |
| 45 | 14 | 8 | 16 | 19 | 17 | 9 | 7 | 15 | 3 | 6 | 10 | 18 | 1 | 2 | 5 | 13 | 11 | 20 | 4 | 12 |
| 46 | 3 | 20 | 17 | 16 | 18 | 6 | 5 | 15 | 12 | 7 | 1 | 9 | 10 | 11 | 8 | 4 | 13 | 2 | 14 | 19 |
| 47 | 10 | 9 | 8 | 11 | 12 | 15 | 5 | 3 | 2 | 13 | 20 | 19 | 4 | 18 | 14 | 7 | 16 | 1 | 17 | 6 |
| 48 | 14 | 8 | 4 | 13 | 5 | 7 | 9 | 12 | 15 | 17 | 1 | 6 | 16 | 19 | 11 | 20 | 3 | 10 | 18 | 2 |
| 49 | 11 | 2 | 10 | 4 | 13 | 18 | 6 | 12 | 8 | 15 | 7 | 19 | 16 | 14 | 17 | 3 | 5 | 1 | 20 | 9 |
| 50 | 15 | 12 | 11 | 10 | 3 | 7 | 13 | 6 | 14 | 19 | 4 | 9 | 5 | 20 | 8 | 18 | 16 | 2 | 17 | 1 |

## A n h a n g 1:

### Dokumentation des Computerprogramms APR

In diesem Anhang ist das Computerprogramm aufgelistet, das für die numerischen Untersuchungen im 6. Kapitel benutzt wurde. Zunächst wird das Programm in ALGOL 60 dokumentiert und die Dateneingabe beschrieben.

```
'BEGIN'
'COMMENT'
**************************************************
*              APE                               *
*                                                *
*   ABLAUFPLANUNG MITTELS RANDOMZAHLEN           *
*   -         -                -                 *
*                                                *
*      (EIN HEURISTISCHES VERFAHREN)             *
*                                                *
**************************************************;

'INTEGER' GJ,GM;
'INTEGER''ARRAY' TEXT[1:12];

INP 1:
PRINT(
<<PROBLEMBEZEICHNUNG (BIS ZU 48 ZEICHEN):>>
);
INSTR(TEXT,-1,8);
PRINT(
<<ZAHL DER AUFTRAEGE,ZAHL DER MASCHINEN:>>
);
READ(GJ,GM);
```

```
'BEGIN'
'INTEGER' RI,J,M,BRAN,BALP,BALLE,BIAB,
          ALZ,BX,SUZG,SUBM,USCHR,
          ZEIMAX,IZEI,SPAMAX,ISPA,
          BMAX,IBMAX,BAIB;
'REAL' STW,UG,OG,R,RPLO,RPLOO,PROZ;
'INTEGER''ARRAY' P,Q[1:GJ,1:GM],PER[1:GM];
'ARRAY' A[1:GM];
```

```
'REAL' 'PROCEDURE' RANDOM(R);
'COMMENT' **********************************
*          RANDOM(R) AUS <0,1>               *
**********************************************;

'REAL' R;

'BEGIN'
'REAL' H,H1;
H1:=H:=ABS(LN(ABS(R)));
H:=H+.05;
H:=ENTIER(H*10)/10;
H:=ABS(H1-H)*20;
RANDOM:=ENTIER(H*1E8)/1E8;
'END' RANDOM **********************************;
```

```
'REAL' 'PROCEDURE' BAZ(R,UG,OG,RI);
'COMMENT' **********************************
* BAZ ERZEUGT EIN ELEMENT DER BEARBEI-    *
* TUNGSZEITMATRIX .                       *
* UG:= UNTERGRENZE DER BEARBEITUNGSZEIT   *
* OG:= OBERGRENZE          --!!--         *
* RI >0:  GANZZAHLIGER WERT               *
* RI<=0:  REELLER WERT                    *
**********************************************;

'INTEGER' RI;
'REAL' R,UG,OG;

'BEGIN'
'REAL' M;
R:=RANDOM(R);
'IF'RI>0'THEN'
'BEGIN'
M:=OG-UG+1;
BAZ:=ENTIER(UG+M*R);
'END'
'ELSE'
BAZ:=UG+R*(OG-UG);
'END' BAZ ********************************;
```

```
'PROCEDURE' PERM(R,N,A,P);
'COMMENT' **********************************
* PERM ERZE GT EINE ZEILE DER              *
* MASCHINENFOLGEMATRIX, SODASS             *
* P[1]...P[N] EINE ZUFAELLIGE              *
* PERMUTATION DER ZAHLEN 1...N             *
* IST                                      *
**********************************************;

'INTEGER' N;
'REAL' R;
'INTEGER''ARRAY' P;
'ARRAY' A;

'BEGIN'
'INTEGER' I,K,IP;
'REAL' AA;

'FOR'I:=1'STEP'1'UNTIL'N'DO'
'BEGIN'
R:=RANDOM(R);
A[I]:=R;
P[I]:=I;
'END' INIT;

'FOR'I:=1'STEP'1'UNTIL'N-1'DO'
'FOR'K:=1'STEP'1'UNTIL'N-I'DO'
'BEGIN'
'IF'A[I]>A[I+K]'THEN' 'GOTO' GH;
AA:=A[I];
A[I]:=A[I+K];
A[I+K]:=AA;
IP:=P[I];
P[I]:=P[I+K];
P[I+K]:=IP;

GH:
'END' I,K;
'END' PERM **********************************;
```

```
OUTPUT(4,<<///
<<STARTWERT FUER P UND O       >>,
.9D,//
<<STARTWERT FUER PLAN I        >>,
.9D,//
<<MAXIMALZAHL DER>>,/,
<<ZU ERZEUGENDEN PLAENE        >>,
9ZD,//
<<SUMME DER BEARBEITUNGSZEITEN>>,
9ZD,//
<<GESAMTBELEGUNGSZEIT>>/,
<<IM EIN-MASCHINENFALL         >>,
9ZD,//
<<ZEILENMAXIMUM>>,/,
<<UNTERE SCHRANKE FUER DMAX    >>,
9ZD,/,
<<ERGIBT SICH FUER AUFTRAG     >>,
9ZD,//,
<<SPALTENMAXIMUM>>/,
<<UNTERE SCHRANKE FUER BMAX    >>,
9ZD,/,
<<ERGIBT SICH FUER MASCHINE    >>,
9ZD,//,
<<MAXIMALE BELEGUNGSZEIT>>,/,
<<IM EIN-MASCHINENFALL         >>,
9ZD,/,
<<ERGIBT SICH FUER MASCHINE    >>,
9ZD,//,
<<GROESSTE GEFUNDENE UNTERE>>,/,
<<SCHRANKE FUER DIE ZYKLUSZEIT>>,
9ZD,*>>,
STW,RPLOO,ANZ,SUZS,SUBM,ZEIMAX,
IZEI,SPAMAX,ISPA,BMAX,IBMAX,
USCHR);
'END' KONTROLLDRUCK **********************;
```

```
'PROCEDURE' PQ;
'COMMENT' **********************************
* PQ ERZEUGT DIE MATRIZEN                    *
* P[J,M] : BEARBEITUNGSZEITMATRIX UND        *
* Q[J,M] : MASCHINENFOLGEMATRIX .            *
* J:=1(1)GJ = ZAHL DER AUFTRAEGE             *
* M:=1(1)GM = ZAHL DER MASCHINEN             *
**********************************************;

'BEGIN'
'FOR'J:=1'STEP'1'UNTIL'GJ'DO'
'FOR'M:=1'STEP'1'UNTIL'GM'DO'
P[J,M]:=BAZ(R,UG,OG,R));

'FOR'J:=1'STEP'1'UNTIL'GJ'DO'
'BEGIN'
PERM(R,GM,A,PER);
'FOR'M:=1'STEP'1'UNTIL'GM'DO'
Q[J,M]:=PER[M];
'END' Q;
'END' P-Q ERZEUGUNG ********************;


'PROCEDURE' KDR;
'COMMENT' * ***************************
*          KONTROLLDRUCK                 *
**********************************************;

'BEGIN'
OUTPUT(4,<<*>>);
OUTSTR(TEXT,-1,4);
OUTPUT(4,<<//,<<BEARBEITUNGSDAUER
 P[J,M] DES >>,/,<<AUFTRAGS J  AUF
 MASCHINE M:>>,//>>);
OUTPUT(4,<< <<      M>> >>);
'FOR'M:=1'STEP'1'UNTIL'GM'DO'
OUTPUT(4,<<ZZD>>,M);
OUTPUT(4,<</,<<  J>>,/>>);
'FOR'J:=1'STEP'1'UNTIL'GJ'DO'
'BEGIN'
OUTPUT(4,<</,ZZD,3B>>,J);
'FOR'M:=1'STEP'1'UNTIL'GM'DO'
OUTPUT(4,<<ZZD>>,P[J,M]);
'END'BEARBEITUNGSZEITMATRIX;

OUTPUT(4,<<////,B,<<
MASCHINENFOLGEMATRIX Q[J,M]:
>>,//,<<      M>> >>);
'FOR'M:=1'STEP'1'UNTIL'GM'DO'
OUTPUT(4,<<ZZD>>,M);
OUTPUT(4,<</,<<  J>>,/>>);
'FOR'J:=1'STEP'1'UNTIL'GJ'DO'
'BEGIN'
OUTPUT(4,<</,ZZD,3B>>,J);
'FOR'M:=1'STEP'1'UNTIL'GM'DO'
OUTPUT(4,<<ZZD>>,Q[J,M]);
'END' KDR;
```

```
'PROCEDURE' MINI(FELD,UG,OG,MIN,IMIN);
'COMMENT' **********************************
MIN:= MINIMUM(FELD[UG]...FELD[OG]),
MIN = FELD[IMIN]
**********************************************;

'INTEGER' UG,OG,MIN,IMIN;
'INTEGER''ARRAY' FELD;

'BEGIN'
'INTEGER' I;

MIN:=FELD[UG];
IMIN:=UG;
'FOR'I:=UG'STEP'1'UNTIL'OG'DO'
'BEGIN'
'IF'MIN>FELD[I]'THEN' 'GOTO' M;
MIN:=FELD[I];
IMIN:=I;

M:
'END' I;

'END' MINI ***************;


'PROCEDURE' MAXI(FELD,UG,OG,MAX,IMAX);
'COMMENT' **********************************
MAX:= MAXIMUM(FELD[UG]...FELD[OG]),
MAX = FELD[IMAX]
**********************************************;

'INTEGER' UG,OG,MAX,IMAX;
'INTEGER''ARRAY' FELD;

'BEGIN'
'INTEGER' I;

MAX:=FELD[UG];
IMAX:=UG;
'FOR'I:=UG'STEP'1'UNTIL'OG'DO'
'BEGIN'
'IF'MAX>FELD[I]'THEN' 'GOTO' M;
MAX:=FELD[I];
IMAX:=I;

M:
'END' I;

'END' MAXI ***************;
```

```
STW:=0;
UG:=1;
OG:=10;
RI:=1;


INP 2:
PRINT(<<
SOLLEN DIE BEARBEITUNGSZEIT- UND MA-
>>);
PRINT(<<
SCHINENFOLGEMATRIX ZUFAELLIG ERZEUGT
>>);
PRINT(<<
WERDEN : (JA-1,NEIN=0):
>>);
READ(BRAN);
'IF'BRAN>0'THEN'
'BEGIN'
PRINT(<<
STARTWERT FUER ZUFALLSZAHLENGENERATOR:
>>);
PRINT(<<(AUS DEM INTERVALL <0,1> )>>);
READ(STW);
R:=STW;
PQ;
'END'
'ELSE'
'BEGIN'
PRINT(<<
BEARBEITUNGSZEITMATRIX (ZEILENWEISE):
>>);
'FOR'J:=1'STEP'1'UNTIL'GJ'DO'
'FOR'M:=1'STEP'1'UNTIL'GM'DO'
READ(P[J,M]);
PRINT(<<
MASCHINENFOLGEMATRIX (ZEILENWEISE):
>>);
'FOR'J:=1'STEP'1'UNTIL'GJ'DO'
'FOR'M:=1'STEP'1'UNTIL'GM'DO'
READ(O[J,M]);
'END' MATRIZENEINGABE;
```

```
INP 3:
PRINT(
<<WELCHE ABLAUFPLAENE SOLLEN ERZEUGT WERDEN:>>
);
PRINT(<<KEINE MEHR    (0)>>);
PRINT(<<SEMIAKTIVE    (1)>>);
PRINT(<<AKTIVE        (2)>>);
PRINT(<<UNVERZOEGERTE(3)>>);
READ(BALP);
'IF'BALP=0'THEN' 'GOTO' ENDAPR;

PRINT(
<<SOLLEN ALLE PLAENE AUSGEGEBEN WERDEN:>>
);
PRINT(
<<(JA=1,NEIN=0):>>);
READ(BALLE);
'IF'BALLE=0'THEN'
'BEGIN'
BTAB:=0;
'GOTO' MANZ;
'END' BALLE=0;

PRINT(
<<IN WELCHER FORM:>>
);
PRINT(
<<AUSFUEHRLICHER PLAN MIT ZEICHNUNG (1)>>
);
PRINT(
<<KURZER PLAN (M:  J  HJM....)      (2)>>);
PRINT(
<<NUR ERGEBNIS (PLNR.-ZYK-GES-RAND) (3)>>);
READ(BTAB);

MANZ:
PRINT(
<<ANZAHL DER PLAENE :>>);
READ(ANZ);

PRINT(
<<STARTWERT AUS <0,1> FUER PLAN 1 :>>);
READ(RPLOO);
PRINT(
<<SOLLEN DIE BOOLSCHEN VARIABLEN X(J,K,M),>>
);
PRINT(
<<DIE DIE AUFTRAGSFOLGEN KENNZEICHNEN,>>
);
PRINT(<<AUSGEGEBEN WERDEN :>>);
PRINT(<<(JA=1,NEIN=0):>>);
READ(BX);
```

```
PRINT(
<<WIRD EIN GUETEMASS VORGEGEBEN ?>>);
PRINT(
<<NEIN                        (0)>>);
PRINT(
<<ZYKLUSZEIT                  (1)>>);
PRINT(
<<GESAMTDURCHLAUFZEIT         (2)>>);
PRINT(
<<GESAMTBELEGUNGSZEIT         (3)>>);
READ(BABB);
'IF'BABB>0'THEN'
'BEGIN'
PRINT(
<<GUETEMASS IN PROZENT:>>);
READ(PROZ);
PROZ:=PROZ/100;
'END'
'ELSE'
PROZ:=0;
```

```
'BEGIN'
'INTEGER' PL,AO,GD,Z,ON,ZZO,IR,NZZO,
          DMIN,NJ,K,KK,MN,MAX,HH,
          GBOPT,ZOPT,PLOPTGD,PLOPTZ,
          BOPT,HM,JMIN,
          GB,GBOPT,PLOPTGB;
'REAL' R;
'INTEGER''ARRAY'
N,HA,F,H,D,OP,OM[1:GJ,1:GM],
ZUL[1:GJ,1:2],
SUZEI,HP[1:GJ],
SUSPA,B[1:GM],
GBZ,GDZ,ZZ[1:ANZ];
'ARRAY' RPL[1:ANZ],ROP[1:GJ*GM];
```

```
'INTEGER''PROCEDURE' IRAN(ZZO);
'COMMENT' **************************
* IRAN(ZZO) ERZEUGT EINE GANZZAHLIGE   *
* ZUFALLSZAHL AUS (1...ZZO) ,          *
* WOBEI HIER GILT:                     *
* ZZO = ZAHL DER ZULAESSIGEN           *
*       OPERATIONEN                    *
**************************************;

'INTEGER' ZZO;

'BEGIN'
'COMMENT' *******  R:=RANDOM(R);
IRAN:=ENTIER(R*ZZO)+1;
'END' IRAN ****************************;
```

```
'PROCEDURE' PLOT(PL,BOPT);
'COMMENT' ***************************
* PLOT(PL,BOPT) ERZEUGT ZUM PLAN PL    *
* DAS SOG. GANTTDIAGRAMM .             *
* WENN  .                              *
* BOPT=0 : PLAN.NR. PL                 *
* BOPT=1 : ZYKLUSZEITMINIMUM           *
* BOPT=2 : GESAMTDURCHLAUFZEITMINIMUM  *
* BOPT=3 : BESAMTBELEGUNGSZEITMINIMUM  *
**************************************;

'INTEGER' PL,BOPT;

'BEGIN'
'INTEGER' T,BH,BD;

KOPF(PL,BOPT);
OUTPUT(4,<<///,<<   T>> >>);

'FOR'M:=1'STEP'1'UNTIL'CM'DO'
OUTPUT(4,<< << M>>,3D>>,M);
OUTPUT(4,<</>>);
```

```
'FOR'T:=0'STEP'1'UNTIL'Z'DO'
'BEGIN'
OUTPUT(4,<</,BZZD>>,1);

'FOR'M:=1'STEP'1'UNTIL'GM'DO'
'BEGIN'
BH:=BD:=0;
'FOR'J:=1'STEP'1'UNTIL'GJ'DO'
'BEGIN'
'IF'H[J,M]<T'AND'T<D[J,M]'THEN'
'BEGIN'
OUTPUT(4,<< << *  *>> >>);
'GOTO' INM;
'END' *  *;

'IF'H[J,M]=T'THEN'BH:=J;
'IF'D[J,M]=T'THEN'BD:=J;
'END' J;

'IF'BH>0'THEN'
'BEGIN'
OUTPUT(4,<< << *>>,DD,<<*>> >>,BH);
'GOTO' INM;
'END' AUFTRAGSANFANG;

'IF'BD>0'THEN'
'BEGIN'
OUTPUT(4,<< << ***>> >>);
'GOTO' INM;
'END' AUFTRAGSENDE;

OUTPUT(4,<< <<      >> >>);

INM:
'END'M;

'END'T;
OUTPUT(4,<</>>);

'END'PLOT **********************************;
```

```
'PROCEDURE' KOPF(PL,BOPT);
'COMMENT' **************************
* KOPF ERZEUGT ZU JEDER TABELLE DIE    *
* UEBERSCHRIFT.                        *
**************************************;

'INTEGER' PL,BOPT;

'BEGIN'
'INTEGER' HPL;

'IF'BOPT>0'THEN' OUTPUT(4,<<*>>) 'ELSE'
OUTPUT(4,<<//>>);
OUTSTR(TEXT,-1,4);
'IF'BOPT=1'THEN'
OUTPUT(4,<< //,
<< ZYKLUSZEITOPTIMUM>>,/
<< ********** >> >>);
'IF'BOPT=2'THEN'
OUTPUT(4,<< //,
<< GESAMTDURCHLAUFZEITOPTIMUM>>,/,
<< ********** >> >>);
'IF'BOPT=3'THEN'
OUTPUT(4,<< //,
<< GESAMTBELEGUNGSZEITOPTIMUM>>,/,
<< ********** >> >>);

'IF'BALP=1'THEN'
OUTPUT(4,<</,<< SEMIAKTIVER    >> >>);
'IF'BALP=2'THEN'
OUTPUT(4,<</,<< AKTIVER >> ,7B>>);
'IF'BALP=3'THEN'
OUTPUT(4,<</,<< UNVERZOEGERTER >> >>);

HPL:=PL;
'IF'BOPT=1'THEN' HPL:=PLOPTZ;
'IF'BOPT=2'THEN' HPL:=PLOPTGD;
'IF'BOPT=3'THEN' HPL:=PLOPTGB;

OUTPUT(4,<< <<PLAN NR.>>,3ZD>>,HPL);
OUTPUT(4,<< 2B, <<R:=>>,.9D>>,RPL[HPL]);
OUTPUT(4,<<//,
<< ZYKLUSZEIT          =>>,3ZD,/,
<< GESAMTDURCHLAUFZEIT =>>,3ZD,/,
<< GESAMTBELEGUNGSZEIT =>>,3ZD>>,
ZZ[HPL],GDZ[HPL],GBZ[HPL]);
'END' KOPF **********;
```

```
'PROCEDURE' TAB(PL,BOPT,BTAB);
'COMMENT' ***********************************
* TAB(PL,BOPT,BTAB) ERZEUGT DIE AUSGA- *
* BETABELLE IN ABHAENGIGKEIT VON PL,   *
* BOPT (S. ''PLOT'') UND BTAB :        *
* WENN                                 *
* BTAB=0 : KEINE TABELLE               *
* BTAB=1 : AUSFUEHRLICHER PLAN MIT     *
*          ZEICHNUNG                   *
* BTAB=2 : KURZER PLAN                 *
* BTAB=3 : NUR ERGEBNIS                *
*                                      *
* AUSSERDEM WIRD BERUECKSICHTIGT ,     *
* OB EIN SEMIAKTIVER, AKTIVER ODER     *
* UNVERZOEGERTER PLAN ERSTELLLT WER-   *
* DEN SOLL.                            *
**************************************;

'INTEGER' PL,BOPT,BTAB;

'BEGIN'
'INTEGER''ARRAY' HF[1:CJ];

'IF'BOPT<1'AND'BTAB=0'THEN' 'GOTO' ENDTAB;
'IF'BOPT<1'AND'BTAB>2'THEN'
'BEGIN'
'IF'BALP=1'THEN'
OUTPUT(4,<</,<<SA>> >>);
'IF'BALP=2'THEN'
OUTPUT(4,<</,<< A>> >>);
'IF'BALP=3'THEN'
OUTPUT(4,<</,<<UV>> >>);

OUTPUT(4,<< << PL>>,3ZD,<<  Z=>>,3ZD,
<<  GD=>>,3ZD,<<  GB=>>,3ZD,<<  R=>>,.9D>>,
PL,ZZ[PL],GDZ[PL],GBZ[PL],RPL[PL]);
'GOTO' ENDTAB;
'END' ERGEBNIS PRO PLAN;

KOPF(PL,BOPT);
'IF'BTAB=1'OR'BOPT>0'THEN'
'BEGIN'
OUTPUT(4,<<///,
<<OPNR   N  M  CNM  PNM HANM  FNM>>,
<<  HNM  DNM        >>,/>>);
```

```
'FOR'M:=1'STEP'1'UNTIL'CM'DO'
'BEGIN'
'FOR'J:=1'STEP'1'UNTIL'GJ'DO'
OUTPUT(4,<</,2ZDB,<<*>>,2ZD,2ZD,3ZD,
4ZD,4ZD,4ZD,5ZD,4ZD,B.9D>>,
OP[J,M],J,M,N[J,M],
P[J,M],HA[J,M],F[J,M],H[J,M],D[J,M]);
'COMMENT' ** OUT ** ROP[OP[J,M]] );
OUTPUT(4,<</>>);
'END' M;

PLOT(PL,BOPT);
'GOTO' ENDTAB;
'END' LANGFASSUNG DER AUSGABE;

'IF'BTAB=2'THEN'
'BEGIN'
OUTPUT(4,<<///,5B>>);
'FOR'J:=1'STEP'1'UNTIL'GJ'DO'
'BEGIN'
OUTPUT(4,<< <<   J HJM>>  >>);

'END' J;
'END' UEBERSCHRIFT KURZFASSUNG;

'FOR'M:=1'STEP'1'UNTIL'CM'DO'
'BEGIN'
OUTPUT(4,<</,<< M>>,DD,<<:>> >>,M);

'COMMENT' **********************
* ERZEUGUNG DER ZEITLICHEN REIHEN- *
* FOLGE DER AUFTRAEGE PRO MASCHINE *
**********  **********************;
'FOR'J:=1'STEP'1'UNTIL'GJ'DO'
HF[ F[J,M] ]:=J;
'FOR'J:=1'STEP'1'UNTIL'GJ'DO'
OUTPUT(4,<< ZZDD,3ZD>>,HF[J],H[HF[J],M]);
'END' KURZFASSUNG;

ENDTAB:
'END' TABLEAU **********************;
```

```
STEPO:
SUZS:=SUBM:=0;

'FOR'J:=1'STEP'1'UNTIL'GJ'DO'
'BEGIN'
SUZEI[J]:=0;
'FOR'M:=1'STEP'1'UNTIL'GM-1'DO'
'BEGIN'
M[ J,Q[J,M] ]:=Q[J,M+1];
SUZEI[J]:=SUZEI[J]+P[J,M];
'END' MASCHINE;

M[ J,Q[J,GM] ]:=0;
SUZEI[J]:=SUZEI[J]+P[J,GM];
SUZS:=SUZS+SUZEI[J];
'END' AUFTRAEGE;

'COMMENT' ***********************
SUZS  (SUMME-ZEILEN-SPALTEN)
ENTHAELT DIE SUMME DER BEARBEI-
TUNGSZEITEN
***********************************;

'FOR'M:=1'STEP'1'UNTIL'GM'DO'
'BEGIN'
SUSPA[M]:=0;
'FOR'J:=1'STEP'1'UNTIL'GJ'DO'
'BEGIN'
SUSPA[M]:=SUSPA[M]+P[J,M];
HA[J,M]:=0;

'FOR'HM:=1'STEP'1'UNTIL'GM'DO'
'BEGIN'
'IF'Q[J,HM]'NOTEQUAL'M'THEN'
HA[J,M]:=HA[J,M]+P[ J,Q[J,HM] ]
'ELSE' 'GOTO' WEHA;
'END' HM;

WEHA:
'END'J;


'FOR'J:=1'STEP'1'UNTIL'GJ'DO'
HP[J]:=P[J, ];

'FOR'J:=1'STEP'1'UNTIL'GJ-1'DO'
'FOR'HJ:=J+1'STEP'1'UNTIL'GJ'DO'
'IF'HA[J,M]>HA[HJ,M]'THEN'
'BEGIN'
HH:=HA[J,M];
HA[J,M]:=HA[HJ,M];
HA[HJ,M]:=HH;
```

```
HH:=HP[J];
HP[J]:=HP[HJ];
HP[HJ]:=HH
'END' TAUSCH;

B[M]:=HA[1,M]+HP[1];
'FOR'J:=2'S EP'1'UNTIL'GJ'DO'
B[M]:='IF'B[M]>HA[J,M]
'THEN'B[M]+HP[J]
'ELSE'HA[J,M]+HP[J];

SUBM:=SUBM+B[M];
'END' MASCHINEN;


'COMMENT' **********************
SUZEI[J] (ZEILENSUMME) IST U.S.
         DER DURCHLAUFZEIT VON
         AUFTRAG J
SUSPA[M] (SPALTENSUMME) IST U.S.
         DER BELEGUNGSZEIT VON
         MASCHINE M
B[M]     VERBESSERTE U.S. VON B
**********************************
SUZS     SUMME DER BEARBEITUNGS-
         ZEITEN
SUBM     SUMME DER B[M] IST U.S.
         DER GESAMTBELEGUNGSZEIT
*********************************;


MAXI(SUSPA,1,GM,SPAMAX,ISPA);
MAXI(B    ,1,GM,BMAX   ,IBMAX);
MAXI(SUZEI,1,GJ,ZEIMAX,IZEI);

USCHR:='IF'ZEIMAX>BMAX
       'THEN'ZEIMAX
       'ELSE'BMAX;

'COMMENT' **********************
SPAMAX <= BMAX U.S. FUER DIE MAX.
               BELEGUNGSZEIT =
               ZYKLUSZEIT
ZEIMAX,USCHR.  U.S. FUER DIE ZYK-
               LUSZEIT
*********************************;


KDR;

BOPI:=0;
IPLO:=RPLOO;
```

```
'PROCEDURE' XJKM;
'COMMENT' ***********************************
* XJKM GIBT (AUF WUNSCH) DIE BINAER-    *
* VARIABLEN X(J,K,M) AUS.               *
* OUTPUT-GERAETENUMMER : =GN            *
* (VOREINSTELLUNG : GN=4)               *
***************************************;

'BEGIN'
'INTEGER' GN,Z;
'INTEGER''ARRAY'HF[1:CJ];

GN:=4;
Z:=0;

OUTPUT(GN,<<//>>);
'FOR'M:=1'STEP'1'UNTIL'CM'DO'
'BEGIN'
Z:=0;
OUTPUT(GN,<<'>>);
'FOR'J:=1'STEP'1'UNTIL'CJ'DO'
HF[ F[J,M] ]:=J;
'FOR'J:=1'STEP'1'UNTIL'CJ-1'DO'
'FOR'K:=J+1'STEP'1'UNTIL'CJ'DO'
'BEGIN'
Z:=Z+1;
'IF'Z>08'THEN'
'BEGIN'
Z:=1;
OUTPUT(GN,<<'>>);
'END' NEUE ZEILE;

'COMMENT' **************************
* AUSGABE DER BINAERVARIABLEN IN   *
* DER FORM : << JJKKMM>> ...       *
**************************************;
'IF'F[J,M]<F[K,M]'THEN'
OUTPUT(GN,<<BDD,DD,DD>>,
J,K,M)
'ELSE' Z:=Z-1;
'END' J-K;
'END' MASCHINE;
OUTPUT(GN,<<'>>);
'END' X(JKM) ***************************
*****************************************
*****************************************
*****************************************;
```

```
MISA:
R:=RPLO;
'FOR'PL:=1'STEP'1'UNTIL'ANZ'DO'
'BEGIN'
STEP 1:
'FOR'J:=1'STEP'1'UNTIL'GJ'DO'
'BEGIN'
'FOR'M:=1'STEP'1'UNTIL'GM'DO'
'BEGIN'
F[J,M]:=D[J,M]:=0;
HA[J,M]:=H[J,M]:=OM[J,M]:=-1;
'END';

STEP 2:
MO:=O[J,1];
HA[J,MO]:=H[J,MO]:=0;
OM[J,MO]:=1;
'END' TABLEAUANFANG;


GB:=GD:=Z:=ON:=0;
RPL[PL]:=R;
ZZO:=GJ;

STEP 3:
ON:=ON+1;
HZZO:=0;
ROP[ON]:=R:=RANDOM(R);
DMIN:=999999;

'FOR'M:=1'STEP'1'UNTIL'GM'DO'
'FOR'J:=1'STEP'1'UNTIL'GJ'DO'
'BEGIN'
'IF'OM[J,M]<1'THEN' 'GOTO' INJ1;
D[J,M]:=HA[J,M];
'FOR'HJ:=1'STEP'1'UNTIL'GJ'DO'
'IF'D[HJ,M]>H[J,M]'THEN'
D[J,M]:=D[HJ,M];
'IF'BALP=1'THEN' 'GOTO' INJ1;
'IF'BALP=2'THEN' HH:=H[J,M]+P[J,M];
'IF'BALP=3'THEN' HH:=H[J,M];
'IF'HH<DMIN'THEN'DMIN:=HH;

INJ1:
'END' DMIN-SUCHE;

'FOR'M:=1'STEP'1'UNTIL'GM'DO'
'FOR'J:=1'STEP'1'UNTIL'GJ'DO'
'BEGIN'
'IF'OM[J,M]<1'THEN' 'GOTO'INJ2;
'IF'BALP=1'THEN' 'GOTO' MZIB;
'IF'BALP=2'THEN'
'BEGIN'
'IF'H[J,M]<DMIN'THEN' 'GOTO' MZIB;
'END' AKTIV;
```

```
'IF'BALP=3'THEN'
'BEGIN'
'IF'H[J,M]=DMIN'THEN' 'GOTO' MZIR;
'END' UNVERZOEGERT;

'GOTO' INJ2;

MZIR:
HZZO:=HZZO+1;
ZUL[HZZO,1]:=J;
ZUL[HZZO,2]:=M;

INJ2:
'END' AUSWAHL DER OPERATION;


MKN:
IR:=IRAN(HZZO);
KK:=ZUL[IR,1];
NN:=ZUL[IR,2];
OM[KK,NN]:=0;
OP[KK,NN]:=ON;

MFKN:
MAX:=F[1,NN];
'FOR'J:=2'STEP'1'UNTIL'GJ'DO'
'IF'F[J,NN]>MAX'THEN' MAX:=F[J,NN];
F[KK,NN]:=MAX+1;

MDKN:
D[KK,NN]:=H[KK,NN]+P[KK,NN];

MHAKN:
'IF'N[KK,NN]=0'THEN'
'BEGIN'
ZZO:=ZZO-1;
GD:=GD+D[KK,NN];
'IF'D[KK,NN]>Z'THEN'Z:=D[KK,NN];

'COMMENT' **********************
* AUSGANG NACHDEM DER PLAN VOLL-  *
* STAENDIG IST DURCH ABFRAGE ,OB  *
* ZAHL DER ZULAESSIGEN OPERATIONEN *
* GLEICH NULL (ZZO=0)             *
**********************************;

'IF'ZZO=0'THEN' 'GOTO' END 1;
'END'
'ELSE'
'BEGIN'
MA[MK,N[KK,NN]]:=D[KK,NN];
OM[KK,N[KK,NN]]:=1;
'END';
'GOTO' STEP 3;
```

```
END 1:
GB:=0;
'FOR'M:=1'STEP'1'UNTIL'GM'DO'
'BEGIN'
'FOR'J:=1'STEP'1'UNTIL'GJ'DO'
HP[J]:=D[J,M];
MAXI(HP,1,GJ,MAX,J);
GB:=GB+MAX;
'END' GESAMTBELEGUNGSZEIT;

GBZ[PL]:=GB;
GDZ[PL]:=GD;
ZZ[PL]:=Z;
'IF'ANZ=1'OR'BABB=0'THEN' 'GOTO' MTAB;
'IF'BABB=1'AND'
(Z-USCHR)/USCHR<=PROZ'THEN''GOTO'MABB;
'IF'BABB=2'AND'
(GD-SUZS)/SUZS<=PROZ'THEN''GOTO'MABB;
'IF'BABB=3'AND'
(GB-SUBM)/SUBM<=PROZ'THEN''GOTO'MABB;

MTAB:
TAB(PL,BOPT,BTAB);
'IF'BOPT>0'AND'BX>0'THEN'XJKM;
'END' PL **********************;

'IF'BOPT>0'THEN' 'GOTO'CES;
BABB:=0;

MABB:
'IF'BABB=0'THEN''GOTO'OPTIMUMSUCHE;
ANZ:=PL;
'IF'BABB=1'THEN'
OUTPUT(4,<<*,<<DAS ZYKLUSZEIT>> >>);
'IF'BABB=2'THEN' OUTPUT(4,
<<*,<<DAS GESAMTDURCHLAUFSZEIT>> >>);
'IF'BABB=3'THEN' OUTPUT(4,
<<*,<<DAS GESAMTBELEGUNGSZEIT>> >>);
OUTPUT(4,<< <<OPTIMUM WURDE FUER>> >>);
OUTPUT(4,<</<<PLAN >>,ZZDB,<<GEFUNDEN>>
,/,<<MIT EINER MINDESTGENAUIGKEIT VON>>,
ZZD.DD,<< %>> >>,
PL,'IF'BABB=1'THEN'(Z-USCHR)/USCHR*100
'ELSE''IF'BABB=2'THEN'
(GD-SUZS)/SUZS*100
'ELSE'(GB-SUBM)/SUBM*100);
```

```
OPTIMUMSUCHE:
GDOPT:=GDZ[1];
GBOPT:=GBZ[1];
ZOPT:=ZZ[1];
PLOPTGB:=PLOPTGD:=PLOPTZ:=1;
'FOR'PL:=2'STEP'1'UNTIL'ANZ'DO'
'BEGIN'
'IF'GDZ[PL]<GDOPT'THEN'
'BEGIN'
GDOPT:=GDZ[PL];
PLOPTGD:=PL;
'END' GESAMT-OPTIMUM;

'IF'ZZ[PL]<ZOPT'THEN'
'BEGIN'
ZOPT:=ZZ[PL];
PLOPTZ:=PL;
'END' ZYKLUS-OPTIMUM;

'IF'GBZ[PL]<GBOPT'THEN'
'BEGIN'
GBOPT:=GBZ[PL];
PLOPTGB:=PL;
'END' BELEGUNGSZEITOPTIMUM;

'END' OPTIMUM;

RPLO:=RPL[PLOPTZ];
ANZ:=1;
BOPT:=1;
'GOTO' MIBA;

GES:
'IF'BOPT>2'THEN' 'GOTO' INP3;
BOPT:=BOPT+1;
'IF'BOPT=2'THEN'
RPLO:='IF'PLOPTGD=1'THEN'RPLOO
                        'ELSE'RPL[PLOPTGD];
'IF'BOPT=3'THEN'
RPLO:='IF'PLOPTGB=1'THEN'RPLOO
                        'ELSE'RPL[PLOPTGB];
'GOTO' MIBA;
'END';
'END';

ENDAPR:
'END'
```

Beschreibung des Leistungsumfangs von APR

APR ist ein Programmpaket, um wahlweise semiaktive, aktive oder unverzögerte Ablaufpläne zu erzeugen. Das Programm beginnt zunächst einen Dialog mit dem Benutzer. Es werden folgende Informationen erfragt:

| Daten (Variable im Programm APR) | Bedeutung | Erläuterung |
|---|---|---|
| TEXT | Problembezeichnung. | Es kann ein bis zu 48 Zeichen umfassender Text eingegeben werden. |
| GJ,GM | Zahl der Aufträge, Zahl der Maschinen. | Theoretisch können beliebig große Integer-Zahlen eingegeben werden, die nur durch den verfügbaren Kernspeicher beschränkt sind. |
| BRAN | BRAN hat die Funktion einer Booleschen Variablen. In Abhängigkeit von der Eingabe werden die Matrizen P und Q zufällig erzeugt oder eingelesen. | BRAN kann nur die Werte 0 (Nein) oder 1 (Ja) annehmen. |
| STW | Startwert für den Zufallszahlengenerator. | Der Zufallszahlengenerator wird in der Prozedur RANDOM durch STW initialisiert. |
| BALP | BALP kann die Werte 0,1,2,3 annehmen. | 1 bedeutet die Erzeugung semiaktiver Pläne, 2 die Erzeugung aktiver Pläne, 3 die Erzeugung unverzögerter Pläne. In einem Programmlauf kann sukzessiv die Erzeugung semiaktiver, aktiver und unverzögerter Ablaufpläne geschehen. Der Wert 0 stoppt die Simulationsläufe und beendet das Programm. |

| | | |
|---|---|---|
| BALLE | BALLE hat die Funktion einer Booleschen Variablen, die die Ausgabe steuert. | Ist BALLE gleich Null, so werden nur die „optimalen" Ablaufpläne geplottet und die zugehörigen Tabellen ausgegeben, ansonsten wird jeder erzeugte Ablaufplan geplottet und die erzeugte Tabelle ausgegeben. |
| BTAB | Ist BALLE gleich Eins, so steuert BTAB die Druckausgabe. | Ist BTAB gleich Eins, so wird jeder Ablaufplan geplottet und in Tabellenform ausgegeben; falls BTAB gleich zwei ist, so werden nur die Starttermine der Aufträge bezüglich der Maschinen ausgegeben. Ist BTAB gleich drei, so wird die Nummer des erzeugten Plans ausgegeben, die ermittelten Zielwerte und die Zufallszahl, mit der dieser Ablaufplan erzeugt wurde. |
| ANZ | ANZ gibt die Anzahl der zu erzeugenden Ablaufpläne an. | Im Dialogbetrieb hängt die Größe von ANZ wesentlich von den Variablen GJ und GM ab. |
| RLPO | Zufallszahl aus dem Intervall (0,1). | RLPO steuert die Auswahl aus der Menge der verfügbaren Operationen *SNM* und initialisiert den Zufallszahlengenerator RANDOM und IRAN. |
| BX | BX hat die Funktion einer Booleschen Variablen, die die Binärvariablen $\delta_{I,L,J}$ erzeugt. | Wird BX gleich Eins gesetzt, so werden alle Binärvariablen $\delta_{I,L,J}$, die im MANNE-Modell Eins sind, ausgedruckt. |

| | | |
|---|---|---|
| BABB | BABB ist ein Steuerparameter, der angibt, ob für eine der drei Zielsetzungen ein Gütemaß vorgegeben wird. | Ist BABB größer Null, so wird für eine der drei behandelten Zielsetzungen eine Prozentzahl (Gütemaß) erwartet. |
| PROZ | PROZ ist die Prozentzahl, die der „optimale" Ablaufplan von der errechneten unteren Schranke abweichen darf. Wird ein Plan erzeugt, der diese Bedingung erfüllt, so stoppt das Programm. | Wird kein Ablaufplan gefunden, der die geforderte Güte besitzt, so stoppt das Programm, wenn ANZ Ablaufpläne erzeugt werden. |

Im folgenden ist ein Dialog und das daraus resultierende Beispiel aufgelistet.

```
                    &XBG,BEN=1023801401  HUCKERT,KSB=27,ISB=100,PSB=200&.

Freitag, der  1. Dezember 1978  13.00 Uhr

Auftrag 1731 F15-1 181302 78.11 00.15

GIB KOMMANDOS&:&LFANM.,APR&.

ANGEMELDET: APR(0001.00)  KAT: RO23C1

GIB KOMMANDOS&:&UEB.,APR,ALG60&MO.,&STARTE&.
MO STDHP WURDE ERZEUGT

ENDE PS&ALGOLCOMP (36.01)  6.17

ENDE MONTIERE (22.05)  3.57

START STDHP (1.00)

PROBLEMBEZEICHNUNG (BIS ZU 48 ZEICHEN):&:TESTBEISPIEL     1.12.1978&.

ZAHL DER AUFTRAEGE,ZAHL DER MASCHINEN:&:3,4&.
SOLLEN DIE BEARBEITUNGSZEIT- UND MA-
SCHINENFOLGEMATRIX ZUFAELLIG ERZEUGT
WERDEN : (JA=1,NEIN=0):&:1&.
STARTWERT FUER ZUFALLSZAHLENGENERATOR:
(AUS DEM INTERVALL <0,1> )&:0.654&.
WELCHE ABLAUFPLAENE SOLLEN ERZEUGT WERDEN:
KEINE MEHR    (0)
SEMIAKTIVE    (1)
AKTIVE        (2)
UNVERZOEGERTE(3)&:3&.
SOLLEN ALLE PLAENE AUSGEGEBEN WERDEN:
(JA=1,NEIN=0):&:1&.
IN WELCHER FORM:
AUSFUEHRLICHER PLAN MIT ZEICHNUNG (1)
KURZER PLAN (M:  J  HJM....)      (2)
NUR ERGEBNIS (PLNR.-ZYK-GES-RAND) (3)&:3&.
ANZAHL DER PLAENE :&:10&.
STARTWERT AUS <0,1> FUER PLAN 1 :&:0.324&.
SOLLEN DIE BOOLSCHEN VARIABLEN X(J,K,M),
DIE DIE AUFTRAGSFOLGEN KENNZEICHNEN,
AUSGEGEBEN WERDEN :
(JA=1,NEIN=0):&:1&.
WIRD EIN GUETEMASS VORGEGEBEN ?
NEIN                  (0)
ZYKLUSZEIT            (1)
GESAMTDURCHLAUFZEIT   (2)
GESAMTBELEGUNGSZEIT   (3)&:0&.
```

TESTBEISPIEL 1.12.1978

BEARBEITUNGSDAUER P[J,M] DES
AUFTRAGS J AUF MASCHINE M:

| J \ M | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | 5 | 2 | 5 | 4 |
| 2 | 2 | 4 | 5 | 8 |
| 3 | 4 | 6 | 2 | 2 |

MASCHINENFOLGEMATRIX Q[J,M]:

| J \ M | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | 4 | 3 | 1 | 2 |
| 2 | 4 | 1 | 2 | 3 |
| 3 | 2 | 4 | 3 | 1 |

| | |
|---|---|
| STARTWERT FUER P UND Q | .654000000 |
| STARTWERT FUER PLAN 1 | .324000000 |
| MAXIMALZAHL DER ZU ERZEUGENDEN PLAENE | 10 |
| SUMME DER BEARBEITUNGSZEITEN | 49 |
| GESAMTBELEGUNGSZEIT IM EIN-MASCHINENFALL | 68 |
| ZEILENMAXIMUM | |
| UNTERE SCHRANKE FUER DMAX | 19 |
| ERGIBT SICH FUER AUFTRAG | 2 |
| SPALTENMAXIMUM | |
| UNTERE SCHRANKE FUER BMAX | 14 |
| ERGIBT SICH FUER MASCHINE | 4 |
| MAXIMALE BELEGUNGSZEIT IM EIN-MASCHINENFALL | 19 |
| ERGIBT SICH FUER MASCHINE | 3 |
| GROESSTE GEFUNDENE UNTERE SCHRANKE FUER DIE ZYKLUSZEIT | 19 |

```
UV PL   1  Z=  25  CD=  61  GB=  79  R=.324000000
UV PL   2  Z=  25  CD=  61  GB=  79  R=.136726190
UV PL   3  Z=  26  CD=  74  GB=  88  R=.052241970
UV PL   4  Z=  25  CD=  61  GB=  79  R=.144834870
UV PL   5  Z=  26  CD=  74  GB=  88  R=.502328610
UV PL   6  Z=  25  CD=  61  GB=  79  R=.588963930
UV PL   7  Z=  25  CD=  61  GB=  79  R=.716615520
UV PL   8  Z=  25  CD=  61  GB=  79  R=.494069300
UV PL   9  Z=  26  CD=  66  GB=  88  R=.704114950
UV PL  10  Z=  26  CD=  74  GB=  88  R=.286401070

TESTBEISPIEL    1.12.1978


 ZYKLUSZEITOPTIMUM
 ********/**********
 UNVERZOEGERTER PLAN NR.    1   R:=.324000000

 ZYKLUSZEIT              =  25
 GESAMTDURCHLAUFZEIT     =  61
 GESAMTBELEGUNGSZEIT     =  79

OPNR    N  M  CNM  PNM  HANM  FNM  HNM  DNM

  5  *  1  1    2    5     9    1    9   14
  7  *  2  1    2    2    12    2   14   16
 11  *  3  1    0    4    16    3   16   20

  8  *  1  2    0    2    14    2   14   16
 10  *  2  2    3    4    16    3   16   20
  2  *  3  2    4    6     0    1    0    6

  4  *  1  3    1    5     4    1    4    9
 12  *  2  3    0    5    20    3   20   25
  9  *  3  3    1    2    14    2   14   16

  1  *  1  4    3    4     0    1    0    4
  3  *  2  4    1    8     0    2    4   12
  6  *  3  4    3    2     6    3   12   14
```

Das Programm APR hat ein 3 Aufträge und 4 Maschinen umfassendes Beispiel mit Hilfe der eingelesenen Zufallszahl erzeugt. Es werden zunächst untere Schranken für die optimalen Zielfunktionswerte berechnet. In dem vorliegenden Beispiel liefert das Programm die unteren Schranken:

Z = 19 ZE, GD = 49 ZE, GB = 68 ZE.

Für die Gesamtdurchlaufzeit wird als untere Schranke die gesamte Bearbeitungszeit genommen. Rechnet man mit Algorithmus 7 eine untere Schranke für GD aus, so erhält man einen Wert von 60 ZE. Die optimale Lösung, die mit dem in Anhang 2 dokumentierten Programm gefunden wurde, lautet:

Z = 23 ZE, GD = 61 ZE, GB = 79 ZE.

APR druckt in Kurzfassung die Zielwerte für die 10 zu erzeugenden Ablaufpläne aus. Die Zahlen R sind die benutzten ersten Zufallszahlen, um Operationen, die eingeplant werden sollen, auszuwählen. Unter diesen 10 Plänen wird für jede Zielsetzung der optimale Zielwert gesucht. Das Programm druckt dann den optimalen Plan bezüglich der Zykluszeit aus. Die Größe OPNR gibt dabei die Reihenfolge an, in der Operation IJ (I = 1,...,3; J = 1,...,4) ausgewählt wurde. In der Indexmenge CNM wird nur die auf die Bearbeitung IJ folgende Maschine K angegeben. Die übrigen Größen wurden in Kapitel 3 definiert.

Das Programm „plottet" den „optimalen" Ablaufplan als Gantt-Diagramm. Die Achse T bezeichnet die Zeitpunkte 0,...,25. M001, M002, M003, M004 sind die vier Maschinen.

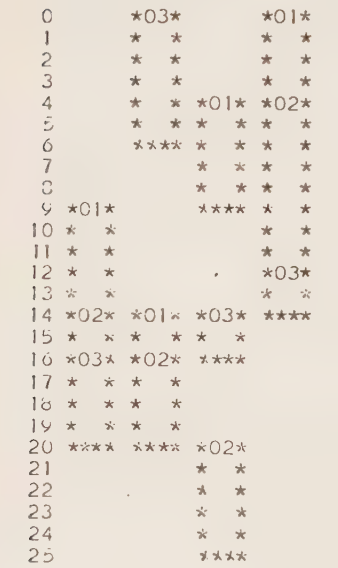
Daraufhin werden diejenigen Binärvariablen $\delta_{I,L,J}$ ausgedruckt, die den Wert 1 haben.

Da zufälligerweise Ablaufplan 1 für alle drei Zielsetzungen den gleichen „optimalen" Ablaufplan erzeugt, wird auf die Wiedergabe des gleichen Ablaufplans für die beiden anderen Zielsetzungen verzichtet.

TESTBEISPIEL 1.12.1978

ZYKLUSZEITOPTIMUM
**********
UNVERZOEGERTER PLAN NR. 1 R:=.324000000

ZYKLUSZEIT = 25
GESAMTDURCHLAUFZEIT = 61
GESAMTBELEGUNGSZEIT = 79




010201 010301 020301
010202
010203 010303
010204 010304 020304

## Anhang 2:

### Dokumentation des MANNE-Modellgenerators

Für das Programm MIXINT von FLEISCHMANN [20] zur Lösung gemischt-ganzzahliger Optimierungsmodelle wurde ein Modellgenerator für die drei behandelten Zielsetzungen erarbeitet. Das Programm führt mit dem Benutzer einen Dialog über die zu lösenden Probleme. Anhand dieses Dialogs erstellt das Programm in der Datei DALP2 die Restriktionen und Zielfunktionen für das gemischt ganzzahlige Optimierungsmodell.

Die einzugebenden Daten entsprechen dabei zum einen dem in Anhang 1 aufgelisteten Computerprogramm, zum anderen der Programmbeschreibung MIXINT aus der Programmbibliothek des Rechenzentrums der Universität des Saarlandes. Aus diesen Gründen wird auf eine ausführliche Dokumentation verzichtet und nur das Programmlisting angegeben.

```
'BEGIN'
'COMMENT' ***************************

     ABLAUFPLANUNG (9.10.78)
     =======================
ALP LIEST ODER ERZEUGT ZUFAELLIG DIE
BEARBEITUNGSZEIT- UND MASCHINENFOLGEMA-
TRIX UND GENERIERT DAS AUSGANGSTABLEAU
IN DER DATEI DALP2 FUER DAS
PROGRAMM ''MIXINT'' (MODUS=2).

*************************************;

'INTEGER' PNR,GJ,GM,RI,IBOUND;
'REAL' STW,UG,OG,BOUND,EPS,G,RESTART;
'BOOLEAN' MAX,DUAL;
'INTEGER' 'ARRAY' TEXT[1:12];


STW:=0;
UG:=1;
OG:=10;
RI:=1;


INP1:
PRINT(<<
PROBLEMBEZEICHNUNG (BIS ZU 48 ZEICHEN):
>>);
INSTR(TEXT,-1,8);
PRINT(<<
PROBLEMNUMMER (GANZE ZAHL):
>>);
READ(PNR);
PRINT(<<
ZAHL DER AUFTRAEGE, ZAHL DER MASCHINEN:
>>);
READ(GJ,GM);

PRINT(<<IST DIE ZIELFUNKTION>>);
PRINT(<<ZU MAXIMIEREN ( 'TRUE')  ODER>>);
PRINT(<<ZU MINIMIEREN ('FALSE') !>>);
READ(MAX);

PRINT(<<DUALWERTE GEWUENSCHT !>>);
PRINT(<<(JA='TRUE' , NEIN='FALSE')>>);
READ(DUAL);
```

```
PRINT(<<
WIRD EINE UNTERE (OBERE) SCHRANKE FUER>>);
PRINT(<<
DAS MAXIMUM (MINIMUM) DER ZIELFUNKTION>>);
PRINT(<<
VORGEGEBEN (IBOUND>0) ODER NICHT (IBOUND=0)!
>>);
PRINT(<<EINGABE IBOUND (GANZE ZAHL):>>);
READ(IBOUND);

PRINT(<<VORGEGEBENE SCHRANKE :>>);
PRINT(<<
(FUER IBOUND=0 OHNE BEDEUTUNG; MUSS>>);
PRINT(<<
ABER TROTZDEM ANGEGEBEN WERDEN.)>>);
READ(BOUND);

PRINT(<<
SCHRANKE EPSILON IM SIMPLEXVERFAHREN:>>);
PRINT(<<(SINNVOLL 1E-5 BIS 1E-8)>>);
READ(EPS);

PRINT(<<
ANZAHL DER HEXADEZIMALSTELLEN, NACH DENEN>>);
PRINT(<<
BEIM PIVOTSCHRITT GERUNDET WIRD:>>);
PRINT(<<
(SINNVOLL: 6 BIS 9)>>);
READ(G);

PRINT(<<
ANZAHL DER SCHRITTE IM BENDERS-VERFAHREN,>>);
PRINT(<<
NACH DENEN DAS SIMPLEXVERFAHREN WIEDER>>);
PRINT(<<
BEI DER ANFANGSMATRIX BEGINNT:>>);
PRINT(<<
(SINNVOLL: 3 BIS 8 BEI GEFAHR VON>>);
PRINT(<<
RUNDUNGSFEHLERN, SONST GROESSER 20)>>);
READ(RESTART);
PRINT(<< >>);


'BEGIN'
'INTEGER' BRAN,BZIFU,Z,J,K,M,II,JJ,ID,
          OXM,OX,OH,OMF,OAF,OZZ,GNO,
          NMX,MMX,NUE,MUEO,MUE1,MUE2,
          OGB,JBM;
'REAL' R,CJJ,AIJ,C,PI;
'INTEGER' 'ARRAY' PER[1:GM],O[1:GJ,1:GM];
'ARRAY' P[1:GJ,1:GM],A[1:GM];
```

```
'REAL' 'PROCEDURE' RANDOM(R);
'COMMENT' ************************************
          RANDOM(R) AUS <0,1>
*************************************************;

'REAL' R;

'BEGIN'
'REAL' H,H1;
H1:=H:=ABS(LN(ABS(R)));
'IF'R<0'THEN'H:=H-.05
       'ELSE'H:=H+.05;
H:=ENTIER(H*10)/10;
R:=ABS(H1-H)*20;
RANDOM:=ENTIER(R*1E8)/1E8;
'END' RANDOM ***********************************;


'REAL' 'PROCEDURE' BAZ(R,UG,OG,RI);
'COMMENT' ************************************
BAZ ERZEUGT EIN ELEMENT DER BEARBEITUNGS-
ZEITMATRIX .
UG:= UNTERGRENZE DER BEARBEITUNGSZEIT
OG:= OBERGRENZE          --!!--
RI >0:  GANZZAHLIGER WERT
RI<=0:  REELLER WERT
*************************************************;

'INTEGER' RI;
'REAL' R,UG,OG;

'BEGIN'
'REAL' M;
R:=RANDOM(R);
'IF'RI>0'THEN'
'BEGIN'
M:=OG-UG+1;
BAZ:=ENTIER(UG+M*R);
'END'
'ELSE'
BAZ:=UG+R*(OG-UG);
'END' BAZ ***************************************;
```

```
'PROCEDURE' PERM(R,N,A,P);
'COMMENT' ***********************
PERM ERZEUGT EINE ZEILE DER
MASCHINENFOLGEMATRIX, SODASS
P[1]...P[N] EINE ZUFAELLIGE
PERMUTATION DER ZAHLEN 1...N
IST
*************************************;

'INTEGER' N;
'REAL' R;
'INTEGER''ARRAY' P;
'ARRAY' A;

'BEGIN'
'INTEGER' I,K,IP;
'REAL' AA;

'FOR'I:=1'STEP'1'UNTIL'N'DO'
'BEGIN'
R:=RANDOM(R);
A[I]:=R;
P[I]:=I;
'END' INIT;

'FOR'I:=1'STEP'1'UNTIL'N-1'DO'
'FOR'K:=1'STEP'1'UNTIL'N-I'DO'
'BEGIN'
'IF'A[I]>A[I+K]'THEN' 'GOTO' GR;
AA:=A[I];
A[I]:=A[I+K];
A[I+K]:=AA;
IP:=P[I];
P[I]:=P[I+K];
P[I+K]:=IP;

GR:
'END' I,K;
'END' PERM ***********************;
```

```
'PROCEDURE' PQ;
'COMMENT' **********************
PQ ERZEUGT DIE MATRIZEN
P[J,M] : BEARBEITUNGSZEITMATRIX UND
Q[J,M] : MASCHINENFOLGEMATRIX .
J:=1(1)GJ = ZAHL DER AUFTRAEGE
M:=1(1)GM = ZAHL DER MASCHINEN
**********************************;

'BEGIN'
'FOR'J:=1'STEP'1'UNTIL'GJ'DO'
'FOR'M:=1'STEP'1'UNTIL'GM'DO'
P[J,M]:=BAZ(R,UG,OG,RI);

'FOR'J:=1'STEP'1'UNTIL'GJ'DO'
'BEGIN'
PERM(R,GM,A,PER);
'FOR'M:=1'STEP'1'UNTIL'GM'DO'
Q[J,M]:=PER[M];
'END' Q;
'END' P-Q ERZEUGUNG **********;
```

```
'PROCEDURE' KDR;
'COMMENT' ********************************
          KONTROLLDRUCK
*********************************************;

'BEGIN'
OUTPUT(4,<<*>>);
OUTSTR(TEXT,-1,4);
OUTPUT(4,<</<<PROBLEMNUMMER:=>>,
B-17ZD>>,PNR);
OUTPUT(4,<<//,<<BEARBEITUNGSDAUER
 P[J,M] DES >>,/,<<AUFTRAGS J  AUF
 MASCHINE M:>>,/>>);
'FOR'J:=1'STEP'1'UNTIL'GJ'DO'
'FOR'M:=1'STEP'1'UNTIL'GM'DO'
OUTPUT(4,<</,6B,<<P[>>,ZD,<<,>>,
ZD,<<]:= >>,ZZD.4D>>,
J,M,P[J,M]);
OUTPUT(4,<<////,B,<<
MASCHINENFOLGEMATRIX O[J,M]:
>>,//,<<      M>> >>);
'FOR'M:=1'STEP'1'UNTIL'GM'DO'
OUTPUT(4,<<ZZD>>,M);
OUTPUT(4,<</,<<  J>>,/>>);
'FOR'J:=1'STEP'1'UNTIL'GJ'DO'
'BEGIN'
OUTPUT(4,<</,ZZD,3B>>,J);
'FOR'M:=1'STEP'1'UNTIL'GM'DO'
OUTPUT(4,<<ZZD>>,O[J,M]);
'END' M;


'IF'STW>0'THEN'
OUTPUT(4,<<///,<< STARTWERT:=>>,
B-D.11DE+3D>>,STW);
OUTPUT(4,<<//,<<
REDUNDANTE SCHRANKE C:=>>,B-D.6DE+3D>>,C);
'IF'BZIFU=1'THEN'
OUTPUT(4,<<//,<<ZYKLUSZEITMINIMIERUNG
>>*>>);
'IF'BZIFU=2'THEN'
OUTPUT(4,<<//,<<
GESAMTDURCHLAUFZEITMINIMIERUNG
>>*>>);
'IF'BZIFU=3'THEN'
OUTPUT(4,<<//,<<
GESAMTBELEGUNGSZEITMINIMIERUNG
>>*>>);
'END' KONTROLLDRUCK **********************;
```

```
'PROCEDURE' NUEID(NUE) ERG:(Z,J,K,M);
'COMMENT' **********************************
NUE->ID ORDNET EINEM NUE AUS[1...MXN]
DIE IDENTIFIKATION ID:=(Z,J,K,M) ZU.
WOBEI
X(J,K,M):= (OJJKKMM)
  H(J,M):= (OJJCOMM)
      Z := (ZOOOOOO) -ZYKLUSZEIT-
   B(M) := (OOOOOMM)
DARGESTELLT WIRD.
**********************************;

'INTEGER' NUE,Z,J,K,M;

'BEGIN'
'INTEGER' IH,SU,I;
'IF'BZIFU=3'AND'NUE>OX+OH'THEN'
'BEGIN'
Z:=J:=K:=0;
M:=NUE-(OX+OH);
'GOTO' ENDE;
'END' GESAMTBELEGUNGSZEIT;

'IF'NUE=OX+OH+1'THEN'
'BEGIN'
Z:=1;
J:=K:=M:=0;
'GOTO' ENDE;
'END' Z;

'IF'NUE>OX'THEN'
'BEGIN'
Z:=K:=0;
IH:=NUE-OX;
M:=(IH-1)'//'GJ;
J:=IH-M*GJ;
M:=M+1;
'GOTO' ENDE;
'END' H;

Z:=0;
M:=(NUE-1)'//'OXM+1;
```

```
SU:=0;
'FOR'I:=GJ-1'STEP'-1'UNTIL'1'DO'
'BEGIN'
SU:=SU+I;
'IF'(NUE-(M-1)*OXM)<=SU'THEN'
'BEGIN'
J:=GJ-I;
K:=GJ-SU+NUE-(M-1)*OXM;
'GOTO' ENDE;
'END' NUE<=SU;

'END' I;

ENDE:
'END' NUE->ID ***********************;



'PROCEDURE' IDNUE(Z,J,K,M) ERG:(NUE);
'COMMENT' **********************
ID->NUE ORDNET EINEM ELEMENT MIT DER
IDENTIFIKATION (Z,J,K,M) DEN INDEX NUE
DER ENTSPRECHENDEN VARIABLEN AUS
MIXINT ZU.
**********************************;

'INTEGER' Z,J,K,M,NUE;

'BEGIN'
'INTEGER' SU,I;

'IF'Z>0'THEN'
'BEGIN'
NUE:=NMX;
'GOTO' ENDE;
'END' Z;

'IF'J=0'THEN'
'BEGIN'
NUE:=OX+OH+M;
'GOTO' ENDE;
'END' GB;

'IF'K>0'THEN'
'BEGIN'
SU:=0;
'FOR'I:=GJ-1'STEP'-1'UNTIL'GJ-J'DO'
SU:=SU+I;
NUE:=(M-1)*OXM+SU-GJ+K;
'GOTO' ENDE;
'END' X(J,K,M);

NUE:=OX+(M-1)*GJ+J;

ENDE:
'END' ID->NUE **********************;
```

```
'PROCEDURE' OCJ;
'COMMENT' ************************************
OCJ SCHREIBT (J,ID,C[J]) IN DATEI NR.2
*************************************************;

'BEGIN'
'FOR'GNO:=4'DO'
OUTPUT(GNO,<</,
B4ZDB,-D,DD,DD,DD,B-D.6DE+3D>>,
JJ,     Z, J, K, M,     CJJ     );
ID:=10C000*Z+10000*J+100*K+M;
'IF'JJ<=0X'THEN'ID:=-ID;
PUTS(2,JJ,ID,CJJ);
'END' OCJ ************************************;


'PROCEDURE' OAIJ;
'COMMENT' ************************************
OAIJ SCHREIBT (I,J,A[I,J]) IN DATEI NR.2
*************************************************;

'BEGIN'
'FOR'GNO:=4'DO'
OUTPUT(GNO,<</,
B4ZD,B4ZD,B-D.6DE+3D3B>>,
 II,  JJ,      AIJ     );
PUTS(2,II,JJ,AIJ);
'END' OAIJ ***********************************;
```

```
INP2:
PRINT(<<
SOLLEN DIE BEARBEITUNGSZEIT- UND MA-
>>);
PRINT(<<
SCHINENFOLGEMATRIX ZUFAELLIG ERZEUGT
>>);
PRINT(<<
WERDEN ! (JA=1,NEIN=0):
>>);
READ(BRAN);
PI:=4*ARCTAN(1);
'IF'BRAN>0'THEN'
'BEGIN'
PRINT(<<
STARTWERT FUER ZUFALLSZAHLENGENERATOR:
>>);
READ(STW);
R:=STW;
PQ;
'END'
'ELSE'
'BEGIN'
PRINT(<<
BEARBEITUNGSZEITMATRIX (ZEILENWEISE):
>>);
'FOR'J:=1'STEP'1'UNTIL'GJ'DO'
'FOR'M:=1'STEP'1'UNTIL'GM'DO'
READ(P[J,M]);
PRINT(<<
MASCHINENFOLGEMATRIX (ZEILENWEISE):
>>);
'FOR'J:=1'STEP'1'UNTIL'GJ'DO'
'FOR'M:=1'STEP'1'UNTIL'GM'DO'
READ(U[J,M]);
'END' MATRIZENEINGABE;

PRINT(<<
ZYKLUSZEITMINIMIERUNG !            (1)
>>);
PRINT(<<
GESAMTDURCHLAUFZEITMINIMIERUNG ! (2)
>>);
PRINT(<<
GESAMTBELEGUNGSZEITMINIMIERUNG ? (3)
>>);
READ(BZIFU);
PRINT(<< >>);
'COMMENT'*******************************
        ENDE DER EINGABE
***************************************;
```

```
C:=0;
'FOR'J:=1'STEP'1'UNTIL'GJ'DO'
'FOR'M:=1'STEP'1'UNTIL'GM'DO'
C:=C+P[J,M];


KDR;

OXM:=ENTIER(GJ*(GJ-1)/2+.05);
OX:=GM*OXM;
OH:=GJ*GM;
OMF:=GJ*(GM-1);
OAF:=GM*GJ*(GJ-1);
OZZ:=GJ;
OGJ:=GJ*GM;
NMX:=OX+OH;
'IF'BZIFU=1'THEN'NMX:=NMX+1;
'IF'BZIFU=3'THEN'NMX:=NMX+GM;
MMX:=OMF+OAF;
'IF'BZIFU=1'THEN'MMX:=MMX+OZZ;
'IF'BZIFU=3'THEN'MMX:=MMX+OGB;



'COMMENT' **********************************
ERZEUGUNG DES AUSGANGTABLEAUS FUER
              MIXINT
**********************************;

'FOR'GNO:=4'DO'
OUTPUT(GNO,<<'SZD20B  >>,PNR);
PUTS(2,PNR);
'FOR'GNO:=4'DO'
OUTPUT(GNO,<<',
ZZDB,ZZDB,FB, FB,  +DB,
4ZDB     ,DE+DB,DB  ,ZD>>,
NMX, MMX, MAX,DUAL,IBOUND,
BOUND,   EPS,  G,  RESTART);
PUTS(2,NMX,MMX,MAX,DUAL,IBOUND,BOUND,
EPS,G,RESTART);
```

```
Z:=-0;
CJJ:=0;
'FOR'NUE:=1'STEP'1'UNTIL'OX'DO'
'BEGIN'
NUEID(NUE,Z,J,K,M);
JJ:=NUE;
OCJ;
'END' X(J,K,M);

'FOR'NUE:=OX+1'STEP'1'UNTIL'OX+OH'DO'
'BEGIN'
JJ:=NUE;
CJJ:=0;
NUEID(NUE,Z J,K,M);
'IF'BZIFU=2'AND'
M=Q[J,GM]'THEN'CJJ:=1;
OCJ;
'END' H(J,M);

'IF'BZIFU=1'THEN'
'BEGIN'
JJ:=NMX;
Z:=1;
J:=K:-M:=0;
CJJ:=1;
OCJ;
'END' Z (ZYKLUSZEITMINIMIERING);

'IF'BZIFU=3'THEN'
'FOR'NUE:=OX+OH+1'STEP'1'UNTIL'NMX'DO'
'BEGIN'
JJ:=NUE;
CJJ:=1;
NUEID(NUE,Z,J,K,M);
OCJ;
'END' GESAMTBELEGUNGSZEIT;

JJ:=Z:-J:=K:=M:=0;
CJJ:=0;
OCJ;
'GOTO' MFB;
```

```
MFB:
'FOR'M:=1'STEP'1'UNTIL'CM-1'DO'
'FOR'J:=1'STEP'1'UNTIL'CJ'DO'
'BEGIN'
IDNUE(O,J,O,M,NUE);
II:=NUE-OX;
JJ:=O;
AIJ:=-P[ J,Q[J,M] ];
OAIJ;
IDNUE(O,J,O,Q[J,M],JJ);
AIJ:=1;
OAIJ;
IDNUE(O,J,O,Q[J,M+1],JJ);
AIJ:=-1;
OAIJ;
'END' MFB;


AFB:
'FOR'M:=1'STEP'1'UNTIL'CM'DO'
'FOR'J:=1'STEP'1'UNTIL'CJ-1'DO'
'FOR'K:=J+1'STEP'1'UNTIL'CJ'DO'
'BEGIN'
IDNUE(O,J,K,M,MUEO);
MUE1:=OMF+2*MUEO-1;
MUE2:=MUE1+1;

II:=MUE1;
JJ:=O;
AIJ:=-P[J,M]+C;
OAIJ;
IDNUE(O,J,K,M,JJ);
AIJ:=C;
OAIJ;
IDNUE(O,J,O,M,JJ);
AIJ:=1;
OAIJ;
IDNUE(O,K,O,M,JJ);
AIJ:=-1;
OAIJ;

II:=MUE2;
JJ:=O;
AIJ:=-P[K,M];
OAIJ;
IDNUE(O,J,K,M,JJ);
AIJ:=-C;
OAIJ;
IDNUE(O,J,O,M,JJ);
AIJ:=-1;
OAIJ;
IDNUE(O,K,O,M,JJ);
AIJ:=1;
OAIJ;
'END' J,K,M;
```

```
'COMMENT' ***********************************
          ZYKLUSZEITRESTRIKTIONEN :
***********************************;
'IF'UZIFU=1'THEN'
'BEGIN'
'FOR'J:=1'STEP'1'UNTIL'CJ'DO'
'BEGIN'
II:=OMF+OAF+J;
JJ:=0;
AIJ:=-P[ J,Q[J,CM] ];
OAIJ;
JJ:=NMX;
AIJ:=-1;
OAIJ;
IDNUE(0,J,0,Q[J,CM],JJ);
AIJ:=1;
OAIJ;
'END' ZYKLUSZEITRESTRIKTIONEN;
'END' ;

'IF'UZIFU=3'THEN'
'FOR'M:=1'STEP'1'UNTIL'CM'DO'
'BEGIN'
IDNUE(0,0,0,M,JBM);
'FOR'J:=1'STEP'1'UNTIL'CJ'DO'
'BEGIN'
II:=OMF+OAF+(M-1)*CJ+J;
JJ:=0;
AIJ:=-P[J,M];
OAIJ;
IDNUE(0,J,0,A,JJ);
AIJ:=1;
OAIJ;
JJ:=JBM;
AIJ:=-1;
OAIJ;
'END' MASCHINE M;
'END' BELEGUNGSZEITRESTRIKTIONEN;
II:=JJ:=0;
AIJ:=0;
OAIJ;
```

```
PNR:=1001;
'FOR'GNO:=4'DO'
OUTPUT(GNO,<</>>,PNR);
PUTS(2,PNR);
'COMMENT' ********************************;
'GOTO' ENDE;

OUTPUT(4,<<*>>);
'FOR'NUE:=1'STEP'1'UNTIL'NMX'DO'
'BEGIN'
NUEID(NUE,Z,J,K,M);
OUTPUT(4,<</,4ZD,4(BCD)>>,
NUE,Z,J,K,M);
'END' NUE->ID TEST;

ENDE:
'END' ABLAUFPLANUNG;
'END'
```

Literaturverzeichnis

[1] AGIN, N.: Optimum Seeking with Branch and Bound. Management Science 13 (1966) B-176 - B-185.

[2] AHO, A., J. HOPCROFT, and J. ULLMANN: The Design and Analysis of Computer Algorithm. Reading, Mass.: Addison Wesley 1974.

[3] ASHOUR, S.: Sequencing Theory. Berlin: Springer 1972.

[4] BAKER, K.R.: Introduction to Sequencing and Scheduling. New York-London-Sydney-Toronto: John Wiley 1974.

[5] BALAS, E.: An Additive Algorithm for Solving Linear Programs with Zero-One Variables. Operations Research 13 (1965) 517-546.

[6] BALAS, E.: Discrete Programming by the Filter Method. Operations Research 15 (1967) 915-957.

[7] BALAS, E.: A Note on the Branch and Bound Principle. Operations Research 16 (1968) 442-445.

[8] BALAS, E.: Machine Sequencing via Disjunctive Graphs: An Implicit Enumeration Algorithm. Operations Research 17 (1969) 941-957.

[9] BENDERS, J.F.: Partitioning Procedures for Solving Mixed Variables Programming Problems. Numerische Mathematik 4 (1962) 238-252.

[10] BERTZISS, A.T.: Data Structures. New York-London: Academic Press 1971.

[11] BRUCKER, P., und W. DINKELBACH: Längste Wege in Netzwerken - Kritische Wege in Netzplänen. Wirtschaftswissenschaftliches Studium 3 (1974) 553-558.

[12] BRUCKER, P.: Zur Komplexität von Scheduling-Problemen. In: J. Kohlas, O. Seifert, P. Stähly und H.J. Zimmermann (Eds.): Proceedings in Operations Research 5. Würzburg: Physica 1976.

[13] BURKARD, R.E.: Methoden der ganzzahligen Optimierung. Wien-New York: Springer 1972.

[14] CHARLTON, J.M., and C.C. DEATH: A Generalized Machine-Scheduling Algorithm. Operational Research Quarterly 21 (1970) 127-134.

[15] CONWAY, R., W. MAXWELL, and L. MILLER: Theory of Scheduling. Reading, Mass.: Addison Wesley 1967.

[16] COOK, S.A.: The Complexity of Theorem-Proving Procedures. Proceedings of the Third Annual ACM Symposium on Theory of Computing 1971.

[17] DIJKSTRA, E.W.: A Note on Two Problems in Connection with Graphs. Numerische Mathematik 1 (1959) 269-271.

[18] DINKELBACH, W.: Ablaufplanung in entscheidungstheoretischer Sicht. Zeitschrift für Betriebswirtschaft 47 (1977) 545-566.

[19] FLEISCHMANN, B.: Eine primale Version des Bendersschen Dekompositionsverfahrens und seine Anwendung in der gemischt-ganzzahligen Optimierung. In: L. Collatz, und W. Wetterling (Hrsg.): Numerische Methoden bei Optimierungsaufgaben. Basel: Birkhäuser 1973.

[20] FLEISCHMANN, B.: Programmbeschreibung MIXINT. Rechenzentrum der Universität des Saarlandes, Saarbrücken.

[21] FLORIAN, M., P. TREPANT, and G. McMAHON: An Implicit Enumeration Algorithm for the Sequencing Problem. Management Science, Series B, 17 (1971) 782-792.

[22] FOX, B.L., J.K. LENSTRA, A.H.G. RINNOOY KAN, and L.E. SCHRAGE: Branching from the Largest Upper Bound. European Journal of Operational Research 2 (1978) 191-194.

[23] GALE, D., H.W. KUHN, and A.W. TUCKER: Linear Programming and the Theory of Games. In: T.C. Koopmans (Ed.): Activity Analysis of Production and Allocation. New York-London: John Wiley 1951.

[24] GIFFLER, B., and G. THOMPSON: Algorithm for Solving Production Scheduling Problems. Operations Research 8 (1960) 487-503.

[25] GONZALES, T., and S. SAHNI: Flowshop and Jobshop Schedules: Complexity and Approximations. Operations Research 26 (1978) 36-52.

[26] GREENBERG, H.H.: A Branch-Bound Solution to the General Scheduling Problem. Operations Research 8 (1968) 352-361.

[27] GUTENBERG, E.: Grundlagen der Betriebswirtschaftslehre. Erster Band, Die Produktion, 18. Aufl. Berlin-Göttingen-Heidelberg: Springer 1971.

[28] HELLER, J., and G. LOGEMANN: An Algorithm for the Construction and Evaluation of Feasible Schedules. Management Science 8 (1961) 168-183.

[29] HOSS, K.: Fertigungsablaufplanung mittels operations-analytischer Methoden. Würzburg: Physica 1965.

[30] HU, T.C.: Ganzzahlige Programmierung und Netzwerkflüsse. München-Wien: Oldenbourg 1972.

[31] HUCKERT, K.: Ein nichtlineares Optimierungsmodell für das Job-Shop-Problem. In: K. Brockhoff, W. Dinkelbach, P. Kall, D.B. Pressmar, und K. SPICHER (Eds.): Proceedings in Operations Research 7. Würzburg: Physica 1978.

[32] JAHN, J.: Sequentieller Innerer-Punkt-Algorithmus zur Lösung nichtlinearer Optimierungsprobleme. Diss., Darmstadt 1978.

[33] JOHNSON, S.M.: Optimal Two- and Three-Stage Production Schedules with Set-Up Times. Naval Research Logistics Quarterly 1 (1954) 61-68.

[34] KARP, R.M.: Reducibility among Combinatorial Problems. In: R.E. Miller, and J.W. Thatcher (Eds.): Complexity of Computer Computation. New York: Plenum Press 1972.

[35] KNUTH, D.E.: The Art of Computer Programming. Vol. III: Sorting and Searching. Reading, Mass.: Addison-Wesley 1973.

[36] KOWALSKY, H.J.: Lineare Algebra, 5. Aufl. Berlin: de Gruyter 1970.

[37] LAND, A., and A. DOIG: An Automatic Method for Solving Discrete Programming Problems. Econometrica 28 (1960) 497-520.

[38] LAGEWEG, B.J., J.K. LENSTRA, and A.H.G. RINNOOY KAN: Job-Shop Scheduling by Implicit Enumeration. Management Science 24 (1977) 441-450.

[39] LAWLER, E.L.: A Pseudopolynomial Algorithm for Sequencing Jobs to Minimize Total Tardiness. In: P.L. Hammer, E.L. Johnson, B.H. Korte, and G.L. Nemhauser (Eds.): Studies in Integer Programming (Annals of Discrete Mathematics 1). Amsterdam-New York-Oxford: North-Holland 1977.

[40] LEMKE, C., and K. SPIELBERG: Direct Search Algorithms for Zero-One and Mixed Integer Programming. Operations Research 15 (1967) 892-914.

[41] LENSTRA, J.K., A.H.G. RINNOOY KAN, and P. BRUCKER: Complexity of Machine Scheduling Problems. In: P.L. Hammer, E.L. Johnson, B.H. Korte, and G.L. Nemhauser (Eds.): Studies in Integer Programming (Annals of Discrete Mathematics 1). Amsterdam-New York-Oxford: North-Holland 1977.

[42] LENSTRA, J.K., and A.H.G. RINNOOY KAN: On the Expected Performance of Branch-and-Bound Algorithms. Operations Research 26 (1978) 347-349.

[43] LIESEGANG, D.G.: Möglichkeiten zur wirkungsvollen Gestaltung von Branch and Bound-Verfahren dargestellt an ausgewählten Problemen der Reihenfolgeplanung. Diss. Köln 1974.

[44] MANNE, A.S.: On the Job-Shop Scheduling Problem. Operations Research 8 (1960) 219-223.

[45] MARIMONT, R.B.: A New Method of Checking the Consistency of Precedence Matrices. Journal of the Association for Computing Machinery 6 (1959) 164-171.

[46] MEHLHORN, K.: Effiziente Algorithmen. Stuttgart: Teubner 1977.

[47] MITTEN, L.G.: Branch-and-Bound Methods: General Formulation and Properties. Operations Research 18 (1970) 24-34.

[48] MURTY, K.G.: Linear and Combinatorial Programming. New York-London-Sydney-Toronto: John Wiley 1976.

[49] NEUMANN, K.: Operations Research Verfahren. Band III: Graphentheorie, Netzplantechnik. München-Wien: Hanser 1975.

[50] NOLTEMEIER, H.: Graphentheorie. Berlin: de Gruyter 1976.

[51] RINNOOY KAN, A.H.G.: Machine Scheduling Problems. Den Haag: Martinus Nijhoff 1976.

[52] ROY, B., and B.G. SUSSMANN: Les Problèmes d'Ordonnancement avec Constraintes Disjonctives. SEMA 1964.

[53] SALKIN, H.M.: Integer Programming. Reading, Mass.: Addison Wesley 1975.

[54] SEELBACH, H.: Ablaufplanung. Würzburg: Physica 1975.

[55] SIEGEL, Th.: Optimale Maschinenbelegungsplanung. Berlin: Schmidt 1974.

[56] SUCHOWITZKI, S.I., und I.A. RADTSCHIK: Mathematische Methoden der Netzplantechnik, 2. Aufl. Leipzig: Teubner 1969.

[57] VEINOTT, A.F., and G.B. DANTZIG: Integer Extreme Points. SIAM Review 10 (1968) 371-372.

[58] ZOUTENDIJK, G.: Mathematical Programming Methods. Amsterdam-New York-Oxford: North-Holland 1976.

[59] KACHIJAN, L.G.: A Polynomial Algorithm in Linear Programming. Doklady Akademii Nauk SSSR 244 (1979) 1093-1096.